The Future Of Mass Storage: Microfloppies And Lasers

COMPUTE!

$2.95
March
1986
Issue 70
Vol. 8, No. 3
$3.75 Canada
02193
ISSN 0194-357X

The Leading Magazine Of Home, Educational, And Recreational Computing

Atari SpeedCalc

A Powerful Spreadsheet
Program Inside For 400/800, XL, XE

Switchbox
Electric Pachinko
For Commodore 64, 128,
Amiga, Atari, Atari ST,
Apple, And IBM PC/PCjr

MultiMemory
For 64 And Apple
Load Several BASIC
Programs At Once

Atari BootStuffer
Fit 10 Boot Programs
On A Single Disk!

IBM Fractal Graphics
Fascinating Images
With A New Math

BASIC Sound
On The Atari ST
How To Make Music
And Sound Effects

Requester Windows
In Amiga BASIC
Add Professional Features
To Your Own Programs

0 71486 02193 03

Flight Simulator II Scenery Disks

The Challenge of Accomplished Flight

With a realism comparable to (and in some ways even surpassing) $100,000 aircraft flight simulators, Flight Simulator II includes full flight instrumentation and avionics, and provides a full-color out-the-window view. Instruments are arranged in the format standard to modern aircraft. All the radios needed for IFR flight are included. Front, rear, left, right, and diagonal views let you look in any direction. Program features are clearly documented in a 96-page Pilot's Operating Handbook.

For training in proper flight techniques, Flight Simulator II includes another 96-page instruction manual, compiled by two professional flight instructors with over 8,000 hours flight time and 12,000 hours of aviation teaching experience. You'll learn correct FAA-recommended flight procedures, from basic aircraft control through instrument approaches. To reward your accomplishments, the manual even includes a section on aerobatic maneuvers.

The Realism and Beauty of Flight

Go sight-seeing over detailed, realistic United States scenery. High-speed graphic drivers provide an animated out-the-window view in either day, dusk, or night flying modes.

Flight Simulator II features over 80 airports in four different scenery areas: New York, Chicago, Seattle, and Los Angeles. Six additional Scenery Disks covering the entire Western half of the United States are now available in IBM and C64/128 disk formats.

Apple and Atari versions will be released soon. Each disk covers a geographical region of the country in detail, and is very reasonably priced.

The Pure Fun of "World War I Ace"

When you think you're ready, you can test your flying skills with the "World War I Ace" aerial battle game. This game sends you on a bombing run over heavily-defended enemy territory. Six enemy fighters will attempt to engage you in combat as soon as war is declared. Your aircraft can carry five bombs, and your machine guns are loaded with 100 rounds of ammunition.

See Your Dealer. Flight Simulator II is available on disk for the Apple II, Atari XL/XE, and Commodore 64/128 computers for $49.95. Scenery Disks for the C64 and IBM PC (Jet or Microsoft Flight Simulator) are $19.95 each. A complete Western U.S. Scenery six-disk set is also available for $99.95. For additional product or ordering information, call (800) 637-4983.

subLOGIC Corporation
713 Edgebrook Drive
Champaign IL 61820
(217) 359-8482 Telex: 206995

Order Line: (800) 637-4983
(except in Illinois Alaska, and Hawaii)

We don't care which computer you own.

We'll help you get the most out of it.

CompuServe puts a world of information, communications, and entertainment at your fingertips.

CompuServe is the world's largest information service designed for the personal computer user and managed by the communications professionals who provide business information services to over one quarter of the FORTUNE 500 companies.

Subscribers get a wealth of useful, profitable or just plain interesting information like national news wires, home shopping and banking, travel and sophisticated financial data. Plus electronic mail, national bulletin boards, forums (special interest groups), and a multichannel CB simulator.

You get games and entertainment, too. Board, parlor, sports, space and educational games. Trivia and the first online TV-style game show played for real prizes.

To buy a CompuServe Subscription Kit, see your nearest computer dealer. To receive our informative brochure or to order direct call or write:

CompuServe®

Consumer Information Service, P O Box 20212
5000 Arlington Centre Blvd , Columbus, OH 43220
800-848-8199 In Ohio Call 614-457-0802

An H&R Block Company

Look for Homework Helper and other Spinnaker products at these fine stores.

ALABAMA
AC-3 Computing Products
Montgomery, AL

ALASKA
Computer Concepts
Eagle River, AK

ARKANSAS
Arkansoft
Harrison, AR

ARIZONA
The Computer Room
Flagstaff, AZ

Mesa Computer Mart
Mesa, AZ

Compushare
Phoenix, AZ

Collegian Computer
Phoenix, AZ

Software City
Tuscon, AZ

CALIFORNIA
Compard
Ridgecrest, CA

Software 1st
Santa Rosa, CA

Computertime
Citrus Heights CA

The Software Place
Fairfield, CA

Coast Computer Center
Costa Mesa, CA

Software Central
Pasadena, CA

R W Christ Corp
Campbell, CA

Calsoft
Oakhurst, CA

Dublin Computers
Dublin, CA

Brown Knows Computing
Redlands, CA

Software City
San Diego, CA

Personal Electronics
Goleta, CA

CONNECTICUT
Software City
West Hartford, CT

Softown Inc
Danbury, CT

DISTRICT OF COLUMBIA
"UR" Computer/ Software Needs
Washington, D C.

FLORIDA
Software City
Sarasota, FL

Microline
Fort Lauderdale, FL

Personal Computer Store
Coral Gables, FL

Discount Software
W Palm Beach, FL

Sunshine Discount Software
Fort Lauderdale, FL

Software Cellar
Fort Lauderdale, FL

GEORGIA
Software House
Jonesboro, GA

Software City
Sandy Springs, GA

HAWAII
Keystone Computer Center
Kaneohe, HI

Microcomputer Systems
Honolulu, HI

ILLINDIS
Save on Soltware
Lombard, IL

Softwaire Centre
Naperville, IL

Softwaire Centre
Niles, IL

Computers Plus
Chicago, IL

Peripherals Plus Ltd
Champaign, IL

Kappels Computer Store
Belleville, IL

Highland Computer
Highland, IL

A Byte Better
Rocklord, IL

Ideal Computer Systems
Kankakee, IL

Farnsworth Computer Center
Aurora, IL

Arcola Software Inc
Arcola, IL

Integrated Computer Systems
Macomb, IL

Midwest Inlormation Systems
Galesburg, IL

Alpine Computer Center
Rockford, IL

The Computer Store
Rocklord, IL

INDIANA
Burkat Computer Center
South Bend, IN

The Game Preserve
Indianapolis, IN

Micro Age Computer Store
Indianapolis, IN

KANSAS
Software City
Overland Park, KS

LOUISIANA
Delta Computers
Alexandria, LA

MARYLAND
Software City
Gathersburg, MD

The Program Store
Kensington, MD

MASSACHUSETTS
Orchard Computer
Hyannis, MA

Software City
West Springlield, MA

Feranti - Dege
Boston, MA

The Computer Center
Hanover, MA

General Computer Store
Hanover Mall Area
Framingham, MA

On-Line Computer Systems
Andover, MA

Orchard Computer
Hyannis, MA

The Whiz Computer Stores
Westboro, MA

Land of Electronics
Saugus, MA

MICHIGAN
Micro Station
Southfield, MI

Micro Station
Troy, MI

Retail Computer Center
Farmington Hills, MI

Retail Computer Center
Birmingham, MI

Retail Computer Center
Garden City, MI

Inacomp Computer
Dearborn, MI

Learning Center Limited
Ann Arbor, MI

Micro Key
Fenton, MI

Software Plus of Breton
Village Mall
Grand Rapids, MI

Krums Computer Center
Battle Creek, MI

Computer Talk
Rochester, MI

Software Trends
Clawson, MI

Creative Computers
Grand Haven, MI

Software Library
Keego Harbor, MI

Computers Today
Holland, MI

MINNESDTA
Computer 1 Inc
Baxter, MN

Computer 1 Inc
Bemidj, MN

Northwoods Computers
Detroit Lakes, MN

MISSOURI
Software City
St Louis, MO

Software To Go
St Louis, MO

NEBRASKA
The Computer Works
Bellevue, NB

Computer Connection
Scottsbluff, NB

Software City
Omaha, NB

NEVADA
Software City
Las Vegas, NV

NEW HAMPSHIRE
Systematic Solutions
Amherst, NH

NEW JERSEY
Computerland
Silo Shopping Center
Northfield, NJ

The Program Store
Eatontown, NJ

Yudins TV Inc.
Wyckoft, NJ

Wolfsons Inc
East Orange, NJ

The Program Store
Wayne, NJ

Computerland
Somerville, NJ

Software City
Ridgefield, NJ

Village Computer
Cedar Knolls, NJ

Software City
Pompton Lake, NJ

NEW YORK
Riester's Computer Store
Auburn, NY

Computerland
Little Neck, NY

Micro Images Industries
Flushing, NY

Software Seller
Harrison, NY

Software City
Tonawanda, NY

Software Plus
Albany, NY

Sound Software
Salt Point, NY

Compucon
Smithtown, NY

NDRTH CARDLINA
The Computer Store
Laurinburg, NC

Computer Alternatives
Asheville, NC

NDRTH DAKOTA
Ultra Inc
Bismark, MD

Computer 1 Inc
Fargo, ND

OHIO
Diskcount Software
Columbus, OH

Chucks Computers
Massillon, OH

The Program Store
Columbus, OH

Computer Renaissance
Columbus, OH

Software City
Youngstown, OH

Wyse Book and Office Supply
Archbold, OH

Holcombs
Cleveland, OH

Disk Drive
Toledo, OH

Liberty Computer Services
Bellefontaine, OH

Tech 2000 Micro Computer
Springlield, OH

Computerworld
Alliance, OH

OREGON
The Users Corner
Medford, OR

PENNSYLVANIA
Country Computing
Summit, PA

De Re Computers
Harrisburg, PA

The Computing Source
West Reading, PA

Downington Computer Center
Downington, PA

RHDDE ISLAND
Microlimits
Smithfield, RI

Software Connection
Warwick, RI

SOUTH CAROLINA
Software Haus
Charleston, SC

C C L Software
Charleston, SC

Software Solutions
Westwood Plaza
Charleston, SC

Byte Shop
Columbia, SC

TENNESSEE
Opus 2
Memphis, TN

MCS
Knoxville, TN

TEXAS
Software Place
Houston, TX

The Software Place
Webster, TX

Norton Brothers Computer Center
El Paso, TX

Software City
Austin, TX

Software Store
San Antonio, TX

Software Ink
Wichita Falls, TX

VIRGINIA
Computerland ol Norfolk
Norlolk, VA

Jack Hartman & Co
Roanoke, VA

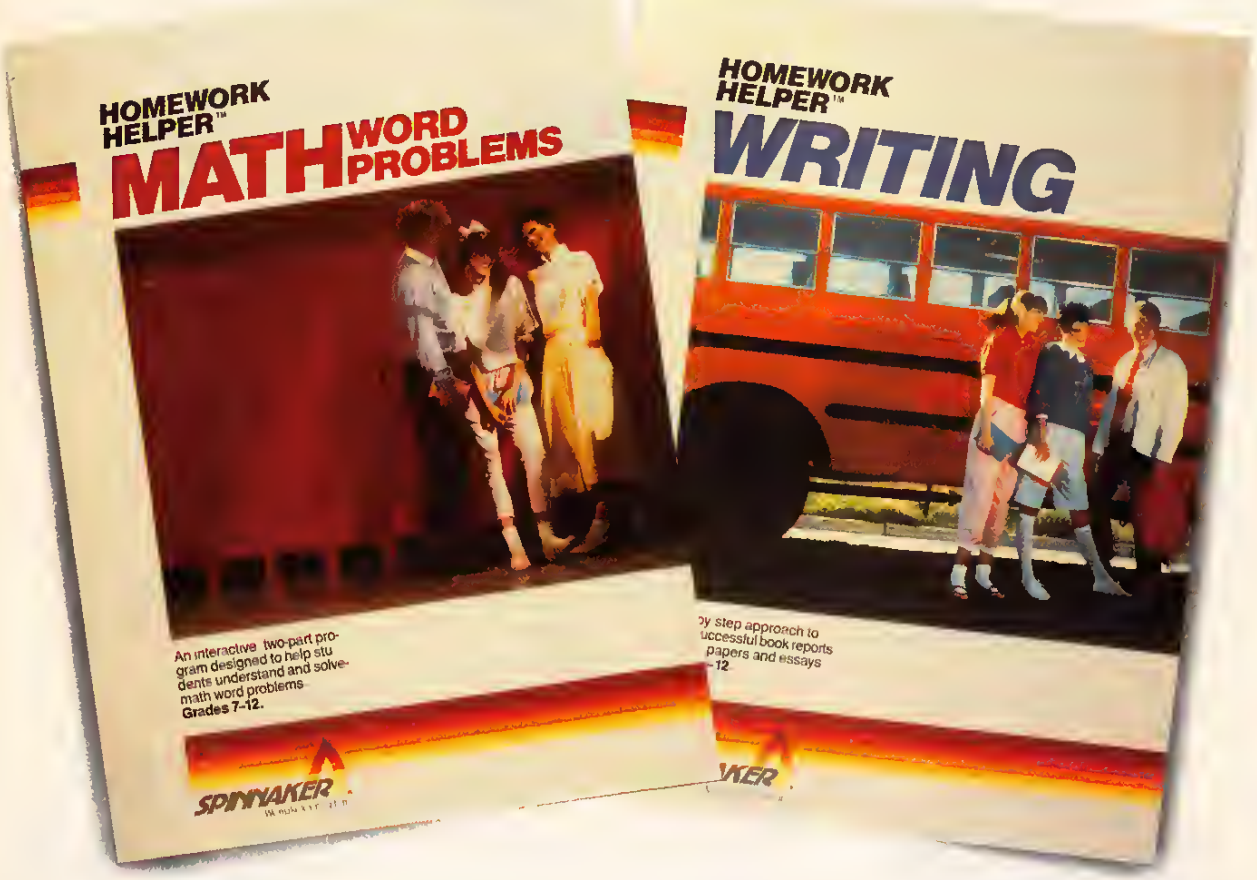
Software Center
Vienna, VA

Family Computer Center
Fairlax, VA

Computerland
Winchester, VA

Software City
Richmond, VA

BLOCK BUSTERS.



Ever run into a mental block while doing your homework? Like how to start an essay? Or solve a tough math problem?

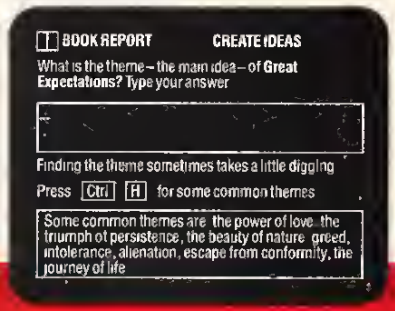
Well now there's help. The HOMEWORK HELPERS™ from Spinnaker. They're designed to bust these blocks and help you produce your homework assignments.

Through a unique system of prompts, the HOMEWORK HELPERS ask stimulating questions, inspire new ideas, and help you organize. And they cover two of the toughest homework challenges: writing well and solving math word problems.

Take WRITING. It has a unique, 3-part program for developing essays and book reports. It gets you started by asking key questions and shows how to organize an outline. Then its built-in word processor actually helps you write, edit, and print out the final work.

Then there's MATH WORD PROBLEMS. Its special grid system helps translate word problems into workable equations. It has a built-in calculator which shows the solution step-by-step and prints out homework calculations ready to hand in.

The Spinnaker HOMEWORK HELPERS are the mental block busters that produce homework results. Look for them at your local software retailer.



SPINNAKER®
We make learning fun.

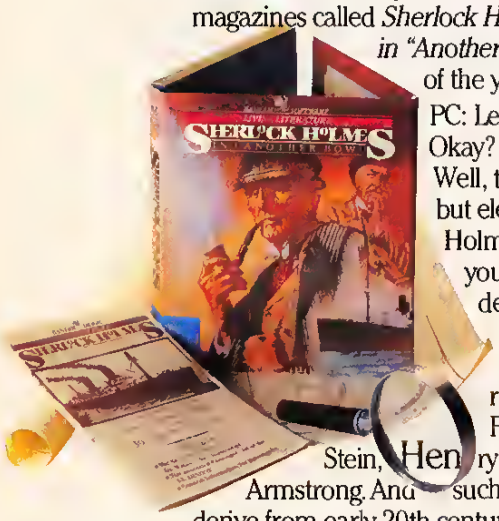
How to turn your computer on.

(The following is an actual conversation between Bantam Software and an unusually talkative personal computer).

BANTAM SOFTWARE: We always ask what turns *people* on. Now we want to know what turns *you* on.

PERSONAL COMPUTER: It's about time someone asked the real expert. What turns me off is boring software. Boring, uninvolving, predictable software. And cold rooms. Why is it always so cold in here?

B: *Games* and *Ahoy* magazines called *Sherlock Holmes in "Another Bow"* one of the year's best.



PC: Let me decide. Okay? (Disk inserted.) Well, this is anything but elementary. You're Holmes. Watson's at your side. And you determine your own fate in case after case. And look, you run into the likes of Picasso, Gertrude Stein, Henry Ford, Louis Armstrong. And such graphics! These derive from early 20th century photographs. I don't have a clue how you did it, but you have a winner. Next case.

B: *The Fourth Protocol*, from Frederick Forsyth's gigantic best-selling book. *Games* called it "nerve-tingling." Here you go. (Slides disk in.)

PC: You mean circuit-tingling. If I knew I had to save the world, I would have gotten more sleep. All kidding aside, this involves *nuclear* weapons. A British traitor. The KGB. And the subversion of NATO. This is a challenge. Will it help if I read the book? (Loud explosion on screen.) Oh no! Does that mean I lost?

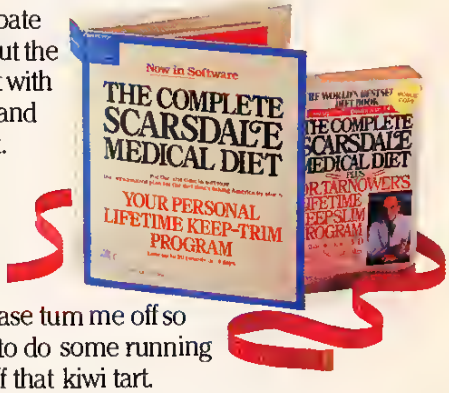
B: No, but losing's the whole point of the next one. *The Complete Scarsdale Medical Diet.* You know the bestseller.

PC: Why, do I look heavy? Never mind, let's have a taste. (Disk is inserted.) This is some menu. It helps you assess your goals. Monitor your progress. Mix 'n match meals from all five Scarsdale diets. Even prepares your shopping list. It'll tell you how much exercise you need to work off certain foods. Let's see about kiwi tart…

B: We've got one other program.

PC: No more. I'm exhausted.

B: No…this is a rebate program. Just fill out the coupon and mail it with proof of purchase and you get $5.00 back.



PC: Thank you. That's a nice offer.

B: So, did we turn you on?

PC: Yup. Now, please turn me off so I can rest. I've got to do some running later on to work off that kiwi tart.

Sherlock Holmes available for Apple //Series Commodore 64/128 IBM PC/PCjr Macintosh
Scarsdale Medical Diet available for Apple //Series IBM PC/PCjr
The Fourth Protocol available for Commodore 64/128 Available soon for Apple //Series and IBM PC/PCjr

BANTAM ELECTRONIC PUBLISHING

Bantam Software Special Offer

$5.00 REBATE!

OFFER ENDS APRIL 15, 1986

To receive your $5.00 Rebate, just send a dated cash register receipt, the warranty card from inside the Bantam Software you bought, plus this rebate form with your name and address clearly printed

Mail to **Bantam Books, Inc., Dept. MT, 666 Fifth Ave., New York, NY 10103**

Name _____ Apt # _____

Address _____ State _____ Zip _____

City _____

Terms Purchase must be made between Jan. 15 1986 and April 15 1986 The warranty card and cash register receipt along with this form (or a plain piece of paper that provides all the above information) must be postmarked no later than midnight April 30 1986 Void where prohibited or restricted by law This offer is not available to employees of Bantam Books Inc or their customers PRINTED IN USA.

C 3/86

COMPUTE!

MARCH 1986
VOLUME 8
NUMBER 3
ISSUE 70

FEATURES

REVIEWS

COLUMNS AND DEPARTMENTS

THE JOURNAL

NOTE: See page 124 before typing in programs.

AP Apple, **Mac** Macintosh, **AT** Atori **ST,** Atari ST, **V** VIC-20, **64** Commodore 64, **+4** Commodore Plus/4, **16** Commodore 16, **128** Commodore 128, **P** PET/CBM, **TI** Texas Instruments, **PC** IBM PC, **PCjr** IBM PCjr, **AM** Amiga, *General Interest

COMPUTE! Publications, Inc.
Part of ABC Consumer Magazines, Inc.
One of the ABC Publishing Companies
ABC Publishing, President, Robert G. Burton
1330 Avenue of the Americas, New York, New York 10019

COMPUTE! The Journal for Progressive Computing (USPS: 537250) is published monthly by COMPUTE! Publications, Inc., 825 7th Ave., New York, NY 10019 USA. Phone. (212) 265-8360. Editorial Offices are located at 324 West Wendover Avenue, Greensboro, NC 27408. Domestic Subscriptions: 12 issues, $24. POSTMASTER: Send address changes to **COMPUTE!** Magazine, P.O. Box 10955, Des Moines, IA 50950. Second class postage paid at Greensboro, NC 27403 and additional mailing offices. ISSN 0194-357X.

Editor's Notes

Now that the hubbub is dying down after the introduction of Atari's ST and Commodore's Amiga, those long-awaited, powerhouse, new-generation computers, perhaps it's a good time to reflect on their relative merits. Although not much software is yet available to show them off to best advantage—a few adventure games, utilities, and applications programs so far—some conclusions can already be drawn.

We've been writing and editing Amiga and ST books and articles here for some months, and our staff is already segregating into camps. We've had camps, of course, for years: Apple enthusiasts, Commodore fans, Atari aficionados, IBM devotees, and assorted other, smaller, clusters of allegiance. It all makes for some spirited exchanges on the relative merits of the competing technologies and, we like to think, energizes our writing and programming.

For example, one of the major responsibilities of our programming staff is transporting programs between machines. We'll transport an arcade game with excellent graphics from its original home to several new computers with varying screen, color, sprite, character, and sound capabilities. This sort of thing throws the differences between computers into high relief.

The Amiga and the ST are quite similar in many respects: Each has a 68000 chip; 512K RAM (although the Amiga is advertised as having only 256K RAM, since the rest is reserved for storing the disk-based operating system); 3½-inch disk drive; mouse; windows; pull-down menus; RS-232 port; parallel printer port; and high-resolution color graphics.

The most striking difference, perhaps, is the price: with color monitor and disk drive, the Amiga costs $1,800, $800 more than the ST. For this extra money, you get multiprocessing, which allows you to run more than one program at a time. The Amiga also offers a more complex sound system with four voices in stereo to the ST's three in mono. The Amiga has 640 × 400 and 640 × 200 resolution modes with 16 simultaneous colors, a 320 × 200 mode with 32 colors, and a total palette of 4,096 colors. The ST has a 640 × 400 monochrome mode, a 640 × 200 mode with 4 simultaneous colors, a 320 × 200 mode with 16 colors, and a total of 512 colors.

Thus, some of the specs would favor the Amiga if, for example, you need extraordinary degrees of color or resolution. Some argue that differences between color number 3,067 and 3,068 are extremely difficult to detect and that this palette represents overkill; others disagree. The Amiga has specialized chips dedicated to memory moves, fills, and other graphics and sound techniques. This frees up the 68000 to do other things while graphics are being manipulated (an important consideration on a computer with a bitmapped, graphics-oriented display). On the other hand, the ST allows the 68000 to run somewhat faster than does the Amiga.

An ST disk holds 360K, the Amiga 880K (although double-sided ST drives with 720K are an option). The Amiga has built-in speech synthesis, but the ST has a built-in MIDI interface for controlling external synthesizers and drum machines. The ST has a built-in hard disk interface; the Amiga requires an additional interface.

Many of the differences between the machines can be eliminated, however, by upgrading, adding peripherals, cards, or options. For example, Commodore will offer a plug-in MIDI interface, and, doubtless, speech synthesis will be made available for the ST. Commodore has announced and demonstrated IBM compatibility via a software emulator, opening up a huge software base. Of course, ST developers are likely to be working on this, too.

While not claiming that the COMPUTE! staff represents a microcosm of the computer marketplace, we have heard effective defenses of both computers. One of our ST partisans says that the disk I/O is faster; software is in greater supply; the operating system and hardware have been around longer and are therefore more fully tested; the machine is easier to understand; there's more speed except for graphics-oriented computing; the keyboard is excellent; the debugger is better; nobody needs multitasking (who could stay in control while simultaneously supervising a spreadsheet and calling a bulletin board?); anything you want that the ST doesn't have you can add; and so forth.

An Amiga owner insists that his computer can expect a great deal of software very soon (the ST was released earlier, and much of its current software comes from Europe where the Amiga has yet to be introduced); the Amiga is hardly an untested technology—it's been in development for three years; the difference in the clock speeds is rendered irrelevant because of the layers of systems, software, languages, and applications above a clock; such things as area fill are built into the Amiga hardware which further counters any clock differences; adding on things not built into the machine results in a pile of extra cords and extra expense; built-in speech means that all programs can use that feature without worrying about compatibility; multitasking is quite useful—having more than one program resident in RAM avoids disk-swapping or rebooting and also allows unrelated software to act as if it were an integrated package.

Rising to the occasion, the ST proponent counters that provisions for multitasking are possible in the ST as well. And so it goes.

At user groups, in magazines, and on telecommunications services around the country, advocates urge one another to get realistic and accept the fact that machine A is obviously better than B. Any comparison of them is contrapuntal; any argument designed to demonstrate the superiority of one can be met by an equally convincing counterargument. It's not surprising that this debate has vitality. After all, the COMPUTE! staff has been working closely with many different machines for years and, with rare exceptions, our Atari camp has never been able to convert the Commodore camp and vice versa, not to mention the solidity of Apple, IBM, and other allegiances. It appears that the ST and Amiga have raised new flags and are likely to perpetuate the friendly face-off that's been an energizing force in personal computing for a decade.

Richard Mansfield

Senior Editor

THE SHADOW $89.95

Shadow is a new and revolutionary piece of hardware that is used to duplicate even the most protected software. Fitting inside the disk drive (no soldering required), SHADDW takes complete control of all functions giving near 10D% copies.

Being the best utility available today, it will even copy the other copy programs.

Because of the Shadow's unique abilities, we feel DOS protection is a thing of the past.

*HACKER PACKAGE $39.95

Shadow a disk while it loads, then read an exact list of:

- Track, sector, ID, check sum, drive status
- High and low track limits
- Density use on each track
- Half tracks that are used
- Command recorder shows commands that were sent to 1541 while program was loading
- RAM recorder records custom DOS

Shadow-scan any disk, then read exact list of:

- Valid tracks, half tracks, partial tracks and segments
- Sync mark link, header block links and data block links
- Track to track synchronization

Exclusive snap shot recorder will give you an exact copy of the 1541 RAM and can be viewed, saved or printed. Plus many more features included.

*Requires Shadow

*GT PACKAGE $44.95

Highly sophisticated and integrated piece of hardware that turns you 1541 into something you've always wanted.

- Track and sector display
- Drive reset switch
- Device number change
- Half track indicator
- Abnormal bit density indicator
- Shadow on-off indicator

The Shadow display will give you an accurate display of precisely what track you are accessing during a normal load even if the program does a read past track 35.

*Requires Shadow

Order by phone 24 hrs./7 days or send cashier's check/money order payable to Megasoft. Visa, MasterCard include card number and expiration date. Add $3.50 shipping/handling for continental U.S., $5.50 for UPS air. CODs add $7.50, Canada add $10.00. Other foreign orders add $15.00 and remit certified U.S. funds only. Distributors invited and supported.

MegaSoft LTD

P.O. Box 1080 • Battle Ground, Washington 98604
1-800-541-1541
Canadian/Foreign Orders Call
(206) 687-5205

Publisher	James A. Casella
Founder/Editor In Chief	Robert C. Lock
Director of Administration	Alice S. Wolfe
Senior Editor	Richard Mansfield
Managing Editor	Kathleen Martinek
Executive Editor	Selby Bateman
Editor	Tom R. Halfhill
Assistant Editor	Philip Nelson
Production Director	Tony Roberts
Production Editor	Gail Cowper
Editor, COMPUTE!'s GAZETTE	Lance Elko
Technical Editor	Otis R. Cowper
Assistant Technical Editors	John Krause, George Miller
Program Editor	Charles Brannon
Assistant Editor, COMPUTE!'s GAZETTE	Todd Heimarck
Assistant Features Editor	Kathy Yakal
Programming Supervisor	Patrick Parrish
Editorial Programmers	Tim Victor, Kevin Mykytyn
Research/Copy Editor	Joan Rouleau
Copy Editor	Ann Davies
Submissions Reviewer	Mark Tuttle
Programming Assistants	David Florance, David Hensley
Executive Assistant	Debi Nash
Administrative Assistants	Julia Fleming, Ins Brooks, Mary Hunt, Sybil Agee
Associate Editors	Jim Butterfield, Toronto, Canada Harvey Herman, Greensboro, NC Fred D'Ignazio, Roanoke, VA David Thornburg, Los Altos, CA
Contributing Editor	Bill Wilkinson
COMPUTE!'s Book Division Editor	Stephen Levy
Assistant Editor	Gregg Keizer
Director, Book Sales & Marketing	Steve Voyatzis
Production Manager	Irma Swain
Art & Design Director	Janice R. Fary
Assistant Editor, Art & Design	Lee Noel
Mechanical Art Supervisor	De Potter
Artists	Debbie Bray, Dabney Ketrow
Typesetting	Terry Cash, Carole Dunton
Illustrator	Harry Blair
Director of Advertising Sales	Ken Woodard
Production Coordinator	Kathleen Hanlon
Promotion Assistant	Caroline Dark
Customer Service Manager	Diane Longo
Dealer Sales Supervisor	Orchid Tamayo
Individual Order Supervisor	Judy Taylor
Receptionist	Anita Armfield
Warehouse Manager	John Williams
Data Processing Manager	Leon Stokes
Assistants	Chris Cain, Steve Bowman

James A. Casella, President
R. Steven Vetter, Vice President, Finance and Planning

COMPUTE! Publications, Inc., publishes

COMPUTE!
The Journal for Progressive Computing

COMPUTE!'s GAZETTE

COMPUTE! Books

COMPUTE!'s GAZETTE DISK

COMPUTE!'s Apple Applications Special

Editorial offices	324 West Wendover Avenue Suite 200 Greensboro, NC 27408 USA
Corporate offices	825 7th Avenue New York, NY 10019 212-265-8360
Customer Service	800-346-6767 (In NY 212-887-8525)

Coming In Future Issues

Report From The Winter Consumer Electronics Show

Commodore 64 Key Phantom: A Powerful New Technique For Dynamic Keyboard Programming

***SpeedScript* Fontmaker For Atari 400/800, XL, XE**

Smooth-Scrolling Billboards For IBM PC & PCjr

Adding System Power To Atari ST BASIC

Member
ABC Audit Bureau of Circulations
MPA Magazine Publishers Association

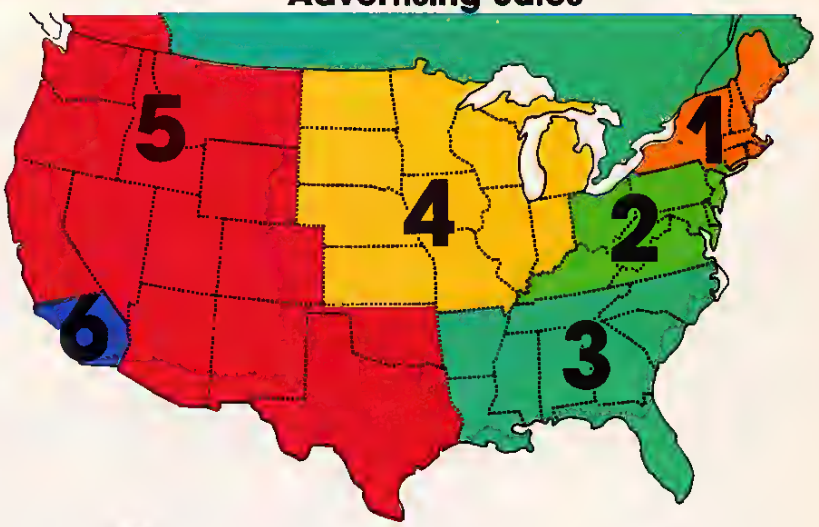
Subscription Orders
COMPUTE!
P.O. Box 10954
Des Moines, IA 50340

TOLL FREE Subscription Order Line
800-247-5470
In IA 800-532-1272

COMPUTE! Subscription Rates (12 Issue Year):

US	(one yr.)	$24
	(two yrs.)	$45
	(three yrs.)	$65
Canada and Foreign Surface Mail		$30
Foreign Air Delivery		$65

Advertising Sales



1. New England
Jonathan M. Just
Regional Manager
212-315-1665

2. Mid Atlantic
John Saval
Eastern Advertising Manager
212-315-1665

3. Southeast & Foreign
Harry Blair
919-275-9809

4. Midwest
Gordon Benson
312-362-1821

5. Northwest/ Mountain/Texas
Phoebe Thompson
Dani Nunes
408-354-5553

6. Southwest
Ed Winchell
213-378-8361

Director of Advertising Sales
Ken Woodard

COMPUTE! Home Office 212-887-8460

Address all advertising materials to:
Kathleen Hanlon
Advertising Production Coordinator
COMPUTE! Magazine
324 West Wendover Avenue
Suite 200
Greensboro, NC 27408

TELECOMPUTING

It's only a phone call away.

MacTalk: Telecomputing on the Macintosh

Sheldon Leemon
Arlan Levitan

A complete guide to telecomputing on the Macintosh from choosing a modem and software to accessing information services and electronic bulletin boards.

$14.95 ISBN 0-942386-85-X

COMPUTE!'s Telecomputing on the IBM

Arlan R. Levitan
Sheldon Leemon

The ins and outs of telecomputing on the IBM PC or PCjr, selecting a modem and evaluating terminal software, how to go online with the major information services.

$14.95 ISBN 0-942386-96-5

COMPUTE!'s Telecomputing on the Commodore 64

Edited

Introduces readers to telecommunications, with sections on buying and using modems, accessing information services and bulletin boards, and uploading and downloading files. There is also a disk available which includes the programs in the book.

$12.95 ISBN 0-87455-009-2

Telecomputing lets you call up computers around the world through a network of telephone lines.

To get you started in telecomputing, COMPUTE! Books offers you five top-selling books. Written for the Apple II-series, Commodore 64, IBM PC and PCjr, and Macintosh, the books give you all the information you need, from selecting software to dialing large databases.

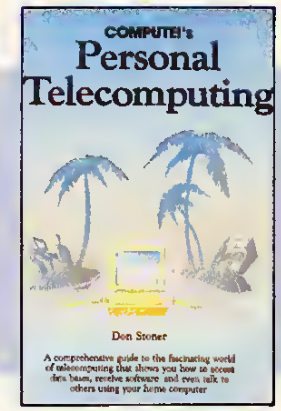
To order your complete guide to telecomputing, give us a call. In the U.S., call toll free 1-800-346-6767 (in NY call 212-887-8525).

COMPUTE!'s Personal Telecomputing

Don Stoner

This comprehensive general guide to the world of telecomputing shows how to access databases, receive software, and communicate with others using a personal computer.

$12.95 ISBN 0-942386-47-7

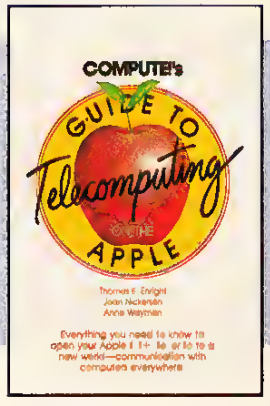


COMPUTE!'s Guide to Telecomputing on the Apple

Thomas E. Enright
Joan Nickerson
Anne Wayman

An informative, easy-to-understand guide to telecomputing on the Apple: covers everything from selecting hardware and software to accessing large databases.

$9.95 ISBN 0-942386-98-1



COMPUTE! Publications, Inc. abc
One of the ABC Publishing Companies

Publishers of COMPUTE! COMPUTE!'s Gazette COMPUTE!'s Gazette Disk COMPUTE! Books and COMPUTE!'s Apple Applications

COMPUTE! books are available in the U.K., Europe, the Middle East, and Africa from Holt Saunders, Ltd., 1 St. Anne's Road, Eastbourne, East Sussex BN21 3UN, England.

Readers Feedback

The Editors and Readers of COMPUTE!

If you have any questions, comments, or suggestions you would like to see addressed in this column, write to "Readers' Feedback," COMPUTE!, P.O. Box 5406, Greensboro, NC 27403. Due to the volume of mail we receive, we regret that we cannot provide personal answers to technical questions.

A Few Helpful REMarks

I suppose this isn't a new idea, but whenever I type in a program printed in COMPUTE!, I add one or two REM lines near the beginning to indicate the program's name, the date, and the page number where it appeared. That way, if I forget some command and can't get the program to work correctly, I can always find the COMPUTE! article which accompanied that program and reread the instructions.

John Hibbs

After a few weeks or months have elapsed, it's easy to forget exactly how a program works, even if it's one you wrote yourself. In most cases, you can't harm a program by adding a couple of REMs. However, you should be careful not to disturb existing lines unless you know exactly what the program is doing. Also, this technique is limited to BASIC programs. Some machine language programs such as Commodore 64 SpeedScript *begin with a line of BASIC (usually something like 0 SYS2061) so that you can load and run the program as if it were BASIC. If you try to add a REM to such a program, it probably won't work at all.*

Computers For Charity

I represent a charitable, nonprofit organization that uses microcomputer equipment in virtually every aspect of its affairs. We would be grateful if your readers would consider contributing additional equipment. Donations of this sort can have substantial financial benefits. If you are in a position to contribute or would like more information, please contact me at the following address or call (617) 495-9020. Collect calls will be accepted.

Robert Epstein, Ph.D.
Executive Director
Cambridge Center For Behavioral Studies
11 Ware Street
Cambridge MA 02138

Virtually every locale has a variety of organizations which may benefit from contributions of computer equipment. Donations may also be tax-deductible. Any readers who want to find out what's available in their area can contact the nearest chapter of the United Way for information about local charitable and volunteer organizations.

Digitized Amiga Sound

In the September 1985 issue of COMPUTE!, you mentioned that the Amiga computer will be able to digitize music. If I were to plug the output from a stereo system or a radio into the Amiga, could it record the music and play it back exactly like the original? How many different voices does the Amiga computer have?

Robert Patterson

The answer to your first question is a qualified yes. The Amiga can play back digitized sound, but you can't plug your stereo output directly into an Amiga and expect to record music without additional hardware and a program to control it. The output from conventional sound equipment is an analog *signal, whereas the Amiga, like other computers, deals only with* digital *information (binary 1's and 0's). Before doing anything else, you'll need to pass the analog signal through an analog-to-digital (A-to-D) converter to put it in a form the computer can use. That sounds more forbidding than it really is: The components for A-to-D converters are cheap and readily available, and it probably won't be long before you see reasonably priced plug-in digitizers for the Amiga.*

Assuming you can convert the incoming signal to digital form, the computer must then sample the signal at a rapid rate—usually thousands of times per second. At each sampling interval, it stores a numeric value which represents the sound input at that point in time. The more frequently you sample the sound, the higher the quality of reproduction—and the more memory is required. The Amiga's 68000 microprocessor runs fast enough to sample incoming signals at an extremely high rate—rivaling the quality of compact disc sound—but even 512K of RAM isn't enough to record significant amounts of high-quality music. Remember that a compact disc can store only up to 75 minutes of music with its capacity of 550 megabytes (563,200K). At that sampling rate, a 512K Amiga could barely record four seconds of music. Of course, by lowering the sampling rate (and accepting somewhat lower quality), that duration can be extended.

At the end of the digital sampling process, the computer has thousands of sample values stored in memory, which can be saved to disk for future use or output directly through a sound channel. To output the digitized sound, you simply reverse the process, reading the stored data from memory, converting it from digital to analog form, and sending the resulting signal to a conventional amplifier at the same rate it was sampled. The Amiga already contains circuitry that can perform the D-to-A conversion at the output end of the process, so sending digitized sound out doesn't require any extra hardware at all.

To answer your second question, the Amiga has four independently programmable sound channels (voices). However, it's difficult to compare them to sound channels on other computers because they're considerably more flexible than tone generators. Most computers are limited to producing one or several basic waveforms, but the Amiga lets you define your own waveforms. And since one channel can modulate (affect) another, it's possible to create extremely sophisticated sounds. Any single channel can simulate a complex waveform, so individual channels can make sounds which would require several channels on other computers. Two of the four channels are assigned to each of the Amiga's stereo outputs, so realistic stereo effects are fairly easy to achieve. The Amiga version of "Switchbox," found elsewhere in this issue, creates stereo effects by switching sounds back and forth between the two stereo outputs.

Moving The New Proofreader's Checksum

When I use the "Automatic Proofreader" with my Commodore 64, the checksum is displayed too high on my screen to be visible. Is there any way to modify the program so it prints the checksum lower on the screen?

Melvin Baral

EPYX PRESENTS LUCASFILM GAMES™ RESCUE ON FRACTALUS!™

© 1985 LFL

You've joined an elite Rescue Squadron, flying to the hostile planet Fractalus to confront the ruthless enemy Jaggies head on. The mission is a treacherous one for, as everyone knows, the cyanitric acid atmosphere on Fractalus is fatal and Jaggi saucers are cunning. You're needed to rescue Ethercorps pilots shot down and stranded on that brutal planet, and to help lead our forces to victory . . . for the merciless Jaggi onslaught must be stopped to preserve the future of our galaxy.

Rescue on Fractalus! is a rescue and space action game with realistic 3-D flight simulation. You pilot your Valkyrie Fighter through the canyons and around the mountain peaks of the planet Fractalus to rescue fellow pilots, do battle with enemy saucers and destroy enemy gun emplacements.

We supply the Long Range Scanner, Dirac Mirror Shield and Anti-Matter Bubble Torpedoes . . . YOU supply the skill and guts! Take the challenge: The perils of Fractalus await you.

	C64/128	ATARI	APPLE
Rescue on Fractalus!	✔	✔	✔

© 1985 LFL

* See specially marked boxes for details. No purchase necessary. Sweepstakes ends Dec 31 1985. Official rules available at participating dealers.

EPYX® COMPUTER SOFTWARE
1043 Kiel Ct., Sunnyvale, CA 94089

Strategy Games for the Action-Game Player®

The problem you mention is typical of TV sets or monitors that suffer from severe overscan (they can't display all of the picture on the screen). If you can't adjust the picture to include the top screen line, you'll have to modify the program. In this issue, we're introducing the "New Automatic Proofreader" for Commodore computers, which works on the 64, 128, VIC-20, Plus/4, and 16. Though it's designed to print the checksum in the upper-left corner of the screen, the new Proofreader can be made to print it elsewhere. First, follow the instructions in the article for typing and saving the new Proofreader. Then reload it and make the following changes:

- ***In line 80, change 20570 to 20551.***
- ***In line 110, change 22054 to 22035.***
- ***In line 190, change 19 to 0.***

Now resave the program, using a different filename so you can tell it apart from the original version. The modified Proofreader prints the checksum just below the last line entered, rather than at the top of the screen. You can either type the next line number over the checksum, or move the cursor down to the next blank line and then start typing. Since this modification makes the Proofreader less convenient for listing and rechecking a group of existing program lines, you probably won't want to make this change unless it's absolutely necessary.

Checking Apple DOS From BASIC

How can an Apple II BASIC program check to see which operating system is running?

P. Nyman

There are quite a few differences between DOS 3.3 and ProDOS, but with a little care, a BASIC program can run under either operating system. You can tell which system is active by PEEKing memory location 48640. This is the start of the BASIC System Global Page in ProDOS, and contains operating system variables which a BASIC programmer might want to read or change. Since the page begins with a machine language JMP (jump) instruction, this first byte has a value of 76 under ProDOS. When you use DOS 3.3, the same byte contains 208. Here's a simple routine that does what you want:

```
10 IF PEEK(48640)=76 THEN PRINT
  "PRODOS INSTALLED":GOTO 30
20 PRINT "DOS 3.3 INSTALLED"
30 REM PROGRAM CONTINUES HERE
```

Atari Compiler Problem

I own an Atari 800XL and frequently use Datasoft's *BASIC Compiler* to compile my own BASIC programs. I recently tried to compile a public domain terminal program called "Amodem 7.1," with unsatisfactory results. The compiler won't accept statements that GOTO or GOSUB a variable or expression. The author of the terminal program used the common memory-saving technique of defining often-used numbers as variables (*C1*=1, and so on). I have converted the variables back to numbers, but the GOTO and GOSUB statements still refer to expressions (for instance, GOTO *3*100* instead of GOTO *C3*C100*). Can you write a routine that will take me the rest of way, or lead me on the right track?

Dennis Brenner

*Since we don't have the terminal program in question, we can't give a specific answer. However, it's not very practical to write a routine that will solve your problem automatically. You'll need to analyze each of the problem statements to determine whether it always branches to the same destination, or branches to different destinations depending on the controlling variable's value. To explain, say that you find the statement GOTO C3*C100 and discover that C3=3 and C100=100. If it's clear that the values of C3 and C100 never change, you can replace the statement with GOTO 300. However, the primary reason for using a variable expression with GOTO or GOSUB is to permit the program to branch to a variety of destinations depending on the variable's value.*

*For example, say that you find the same statement (GOTO C3*C100) and discover that C3 may have the values 1, 2, or 3 when this statement executes. Program flow will branch to line 100, 200, or 300, depending on the value of C3. In this case, you can't replace the expression with a constant, since that would limit the branch to only one destination. The best alternative is to substitute ON-GOTO and ON-GOSUB. For instance, the statement ON C3 GOTO 100, 200, 300 branches to line 100 when C3=1, line 200 when C3=2, and so on. To make this work, you must determine all the possible values that the controlling variable (C3 in this case) might have, and compute all the destinations that might be generated by that expression. Once that's done, you'll know which line numbers to put at the end of the ON-GOTO or ON-GOSUB statement.*

Most BASIC compilers accept only a subset of all the commands in BASIC, so it's possible that yours might not handle ON-GOTO or ON-GOSUB, either. If that's the case, you could replace the original statement with a string of IF-THEN-GOTO statements (IF C3=1 THEN GOTO 100, IF C3=2 THEN GOTO 200, etc.). This construction is less efficient, but should work with almost any compiler. Tradeoffs of this sort are inevitable when compiling programs that weren't designed to be compiled.

Saving IBM PC Screens

I'm writing a BASICA painting program for the IBM PC that does all the drawing with PUT commands. But I need to know how to save a picture to disk so my work isn't lost when I turn the computer off. I know you can store an entire screen in an array with a GET command like this:

```
10 DIM V(4001)
20 GET (0,0)-(639,199),V
```

How can I save the contents of this array to disk?

David Short

*It's a simple operation in BASICA. The VARPTR function can tell you the memory location where any array is stored, and BSAVE can save the contents of any block of memory, including arrays or other variables. Since each element of the array occupies four bytes, you must save 16004 (4*4001) bytes of memory. Use VARPTR(V(0)) to find the location of the first element in the array. This statement saves the array V in a file called PICTURE:*

```
30 BSAVE "PICTURE",VARPTR(V(0))
,16004
```

Here's a complementary program to load the same picture from disk and display it on the screen. Don't forget to DIMension the array before performing this operation.

```
10 DIM V(4001)
20 BLOAD "PICTURE",VARPTR(V(0))
30 PUT (0,0),V,PSET
```

Embedded BASIC Words

I usually pay little or no attention to spacing when typing BASIC programs, but when using "MLX II" to type in a program from the December 1985 issue of COMPUTE!, I ran into a puzzling problem. Everything worked fine until I tried to load data and received the message SYNTAX ERROR IN LINE 830. I rechecked the line and found that everything was correct, except that I hadn't used the same spacing shown in the magazine listing. When I corrected the spacing, the program worked perfectly. After a little further investigation, I discovered that the space causing the problem was between ST and AND. Why does that space make such a difference?

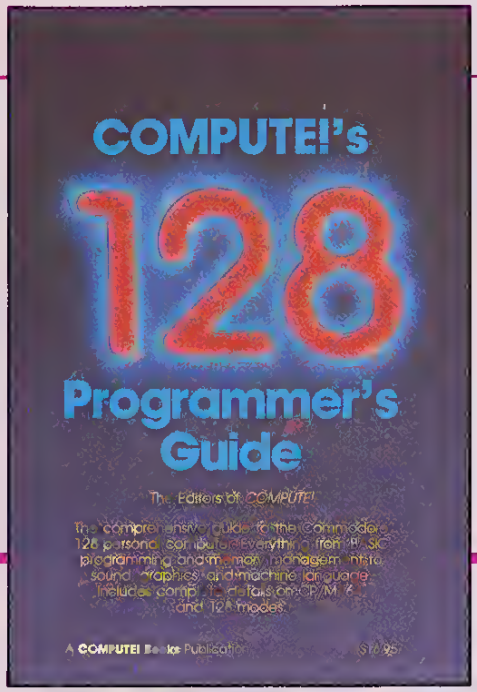
Jim David

Here's what that portion of line 830 looks like:

```
IF ST AND(I<>B)THEN F=2
```

Both AND and ST are reserved words *in Commodore BASIC, meaning that BASIC lets you use them for only one purpose. ST (STatus) is a reserved variable that indicates the status of input/output operations like loading or saving to disk. AND*

COMPUTE!'s PROGRAMMER'S GUIDES



Everything you need for successful, entertaining, and challenging programming on your Amiga, Atari ST, or Commodore 128 computer.

Each book is carefully written in COMPUTE!'s lively, understandable style to help even beginning programmers learn the ins and outs of their personal computers.

COMPUTE!'s 128 Programmer's Guide

ISBN 0-87455-031-9
Editors of COMPUTE! 464 pages

Written and compiled by the most technically proficient authors in consumer computing today, the technical staff of COMPUTE! Publications, this guide to the powerful Commodore 128 computer contains a wealth of information for every programmer. Explore BASIC 7.0 through countless hands-on examples and sample programs. Learn how to create dazzling graphics and sophisticated sounds in both BASIC and machine language. See how to program peripherals, such as disk drives, printers, and modems. Enter the world of CP/M, just one of the three modes of the 128. There are even chapters introducing you to machine language programming and the computer's method of managing memory. *COMPUTE!'s 128 Programmer's Guide* includes numerous appendices covering everything from error messages to memory maps.

$16.95

COMPUTE!'s ST Programmer's Guide

0-87455-023-8
Editors of COMPUTE!

Complete and comprehensive, yet easy to understand, *COMPUTE!'s ST Programmer's Guide* is a must-buy for any Atari ST owner. The technical staff of COMPUTE! Publications has put together a reference guide to programming that takes the reader through every aspect of this newest Atari personal computer. Logo and BASIC, the two programming languages now available for the machine, are explored in detail. From programming concepts to writing programs, the scores of ready-to-type-in examples show just what can be done, and how to do it. Advanced features of this new-generation computer, such as GEM and TOS, the ST's user interface and operating system, are illustrated as readers write their own applications. Valuable appendices provide information programmers need, including GEM VDI opcodes and a list of ST resources.

$16.95

COMPUTE!'s Amiga Programmer's Guide

0-87455-028-9
Edited

Covering AmigaDOS, BASIC, Intuition, and the other important programming tools which accompany the new Amiga, *COMPUTE!'s Amiga Programmer's Guide* is a clear and thorough guide to the inner workings of this fascinating, new-generation computer. The great speed of its 68000 microprocessor, coupled with the versatility of the Amiga-specific graphics and sound chips, makes the Amiga one of the most powerful computers available today. Written by the technical staff of COMPUTE! Publications, the most technically knowledgeable authors in computing today, this book is your key to accessing the Amiga's speed and power.

$16.95 (March Release)

Look for these new books at a bookstore or a computer store near you. Or order directly from COMPUTE! Books. Call toll-free 1-800-346-6767. In NY call 212-887-8525.

COMPUTE! Publications, Inc.
One of the ABC Publishing Companies

COMPUTE! books are available in the U.K., Europe, the Middle East, and Africa from Holt Saunders, Ltd., 1 St. Anne's Road, Eastbourne, East Sussex BN21 3UN, England.

is a logical operator which in this case connects the value of ST with the value of the expression (I<>B).

Typing STAND(I<>B) instead of ST AND(I<>B) makes the computer see a third reserved word in the line—the numeric function TAN (TANgent). Since TAN, like other functions, must be followed by something inside parentheses, the computer responds with a syntax error message when it finds the letter D instead of a left parenthesis. That's a nutshell explanation for the error. But you may still wonder why the computer sees TAN inside the word STAND. After all, the words ST and AND seem to be there as well.

The short program below shows exactly why TAN appears. Don't worry about the fact that line 10 looks strange. We're not going to execute that line—it's only there to let us examine how BASIC handles these reserved words.

```
10 :ST:TAN:AND::STAND::
20 PRINT CHR$(14);CHR$(147)
30 FOR J=0 TO 19:POKE 1024+J,
   PEEK(2049+J):POKE 55296+J,1:NEXT
```

After typing the program, enter GOTO 20 and press RETURN (don't start the program with RUN). Line 30 PEEKs the first 20 bytes of BASIC program space and displays their contents on the screen, showing you how the computer stores line 10 in memory. As you'll see, the reserved variable ST is stored as the ASCII characters S and T, exactly what you typed in. This is the way all variable names are stored. However, both TAN and AND are changed into one-byte tokens, *which appear here as reverse video characters. Most BASIC words are tokenized—compressed into a single numeric value—to save space and make BASIC run faster. Between the double colons we placed in the line as markers, you can see how the computer handles the character sequence S-T-A-N-D. When it tokenizes a BASIC line, the computer reads from left to right, just as you do. The initial S in STAND is left unchanged, since it isn't part of a keyword that can be tokenized. Next, the computer finds the characters T-A-N, which it replaces with the one-character token for TAN. That leaves the character D, which is also left unchanged.*

After TAN is tokenized, the computer can't possibly see ST or AND (T and AN are missing), so the line can't work as intended. In this case, it was coincidental that the combination of two reserved words made a third reserved word. However, the same thing would happen if you omitted a space between ST and the logical operator OR. When the computer scans the characters S-T-O-R, it changes the embedded keyword TO into a token. For similar reasons you should be careful not to use variable names like TOP, NOTE, or FORK, which also contain embedded BASIC words (TO, NOT, and FOR).

Arabian Atari Revisited

In the December 1985 "Readers' Feedback" you printed a letter from Nour Abdullah Al-Rasheed asking how to make the cursor on his Atari computer move from right to left. He may want to consider a hardware solution. The images displayed by a television set or monitor are placed on the screen by vertical and horizontal deflection circuits. An experienced electronics technician who's familiar with video displays should be able to examine the schematic for that device and determine which wires control horizontal deflection. By rewiring that circuit, the technician could bring about the desired change. This modification should probably be considered permanent; and it may require some adjustment of normally untouched internal controls to get a satisfactory picture. While it might be possible to install a switch that would let you flip back and forth between display modes, the technician would have to use special insulating spacers and take pains to protect the operator from the very high voltages involved.

Jim Taylor

Thanks to you and the other readers who suggested this solution. As you point out, the circuitry involved carries extremely high voltages that can cause very serious injury, so this type of modification must be performed by a fully qualified technician. *Unless you fit that category, don't even consider poking around inside your TV or monitor. You may cancel any warranty which is in effect, and run a serious risk of injuring yourself as well as the device.*

Refurbishing Tip

I really appreciate the article "Refurbish Your 64" from the December issue of COMPUTE!. Here is an additional convenience feature. If you change line 3470 to read as follows, you won't have to enter the direct mode statements (POKE 55,0:POKE 56,160:POKE 643,160:POKE 644,160:NEW) after the program is run.

```
3470 READ A0:IF A0=99999 THEN
   POKE253,253:SYS49194:POKE643,0
   :POKE644,160:NEW
```

Albert Alarie

Thanks for the tip.

TI Music

I have seen TI-99/4A programs that create music with DATA statements. Please show me how this is done.

Tim Huemmer

Though the DATA statements play a part in the process, the TI actually makes sound with CALL SOUND. Here's the simplest form of the statement:

CALL SOUND(*d,f,v*)

The first value in parentheses (d) sets the duration *for the sound. The second value (f) sets the* frequency, *and the third (v) sets the* volume. *CALL SOUND lets you produce as many as four tones at once, so with a statement like CALL SOUND (d,f1,v1,f2,v2,f3,v3) it's possible to create a three-note chord. In this case, f1, f2, and f3 represent the frequencies of the three notes, and v1, v2, and v3 represent their respective volumes. Of course, in a program you'd substitute real numbers or variables inside the parameters.*

Where do DATA statements come into the picture? In most cases, it's simplest to read the music data from DATA statements and assign it to variables inside parentheses in CALL SOUND. This saves program space and makes the music data easier to understand and modify. Here's a short example of how it's done:

```
100 V=5
110 FOR I=1 TO 5
120 READ D,F1,F2,F3
130 CALL SOUND(D,F1,V,F2,V,F3,V)
140 NEXT I
150 DATA 1500,262,330,390
160 DATA 250,262,349,440
170 DATA 1500,262,349,415
180 DATA 250,277,349,415
190 DATA 1500,277,370,466
200 DATA 250,262,392,466
```

This program plays five three-note chords. Line 100 assigns the value 5 to the variable V. Since the CALL SOUND statement uses V to set the volume for every note, it stays the same throughout the program. Line 120 READs in new DATA items for each chord, setting the duration with the variable D and the three note frequencies with variables F1, F2, and F3. The frequency values for the notes are found in the appendix in the TI User's Reference Guide. *You can read more about TI sound in* COMPUTE!'s Programmer's Reference Guide to the TI-99/4A *by C. Regena. Several of her monthly columns in* COMPUTE! *have also covered this topic.*

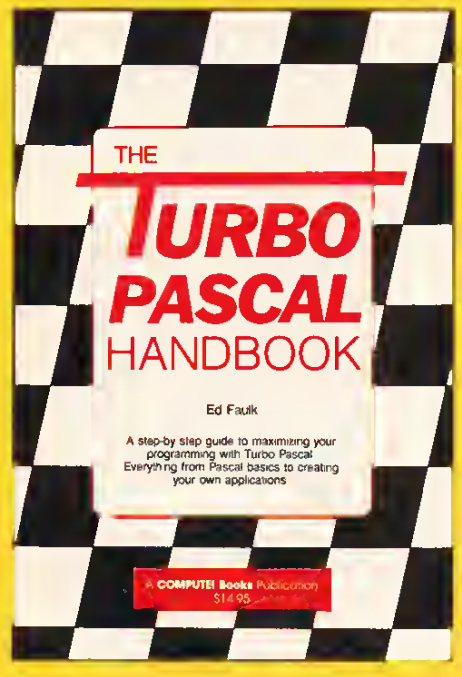
Commodore B128 Users' Group

I was glad to see that Jim Butterfield's dynamic keyboard articles (COMPUTE!, October–December 1985) included some references to the Commodore B128 (called the B700 in Europe). As you may know, the international B128 user group is sending out 13,000 newsletter/membership applications to B128 owners in North America and B700 users in Europe. The group currently has 1,500 members, and membership is rapidly increasing. Our disk library is also off to a good start, and offers a variety of public domain

Explore Pascal with

THE TURBO PASCAL HANDBOOK

from COMPUTE!.



The Turbo Pascal Handbook
Edward P. Faulk
With *The Turbo Pascal Handbook* and *Turbo Pascal* from Borland International, you'll be gently guided, step-by-step, until you're creating your own powerful applications in this impressive computer language.
$14.95 ISBN 0-87455-037-8

This information-packed book from COMPUTE! is an outstanding resource and programming guide. And it's written in COMPUTE!'s bestselling style so that even beginning programmers can quickly and easily understand all the applications.

Ask for *The Turbo Pascal Handbook* at your local computer store or bookstore. Or order directly from COMPUTE!. Call toll free 1-800-346-6767 (in NY 212-887-8525) or mail the attached coupon with your payment (plus $2.00 shipping and handling per book) to COMPUTE! Books, P.O. Box 5038, F.D.R. Station, New York, NY 10150.

Note: You'll need *Turbo Pascal* in order to use this book. The software is not included with *The Turbo Pascal Handbook.*

Yes! Send me _____ copies of *The Turbo Pascal Handbook* at $14.95 each.
My payment is enclosed.

ALL ORDERS MUST BE PREPAID IN U.S. FUNDS

Subtotal _____
NC residents add 4.5% sales tax _____
Shipping and handling _____
($2.00 per book in U.S. and surface mail, $5.00 per book airmail.)
Total enclosed _____

☐ Payment enclosed (check or money order)
☐ Charge ☐ Visa ☐ MasterCard ☐ American Express

Account No. _____ Exp. Date _____
(Required)

Name _____

Address _____

City _____ State _____ Zip _____

Please allow 4–6 weeks for delivery

36303711

COMPUTE!™ Publications, Inc. abc
One of the ABC Publishing Companies
825 7th Avenue, 6th Floor, New York, NY 10019
Publishers of COMPUTE! COMPUTE!'s Gazette COMPUTE!'s Gazette Disk COMPUTE! Books and COMPUTE!'s Apple Applications

programs to members. Interested B128/B700 owners may obtain membership information at the following address:

B128/B700 User's Group
Attn: Norman Deltzke
4102 North Odell
Norridge, Illinois
USA 60634

Thanks to reader John A. Francis for supplying this information.

Booting PCjr In 80 Columns

I own an IBM PCjr, as do many of my coworkers. We would all like to know if there is any way to make the PCjr boot DOS 2.1 in an 80-column format instead of 40 columns. Presently, to get DOS in 80 columns, I execute the program "Rebound" which was published in COMPUTE!. I then press Fn-Break, and the DOS prompt appears in 80 columns. Can you show me a simpler way?

Martin Gappa

No tricks are needed to get 80 columns on the PCjr, since DOS has a command specifically for that purpose. Just type MODE 80 at the DOS prompt with the DOS disk in the drive. To get back to 40 columns, type MODE 40. Additional parameters let you shift the display left or right to center it on the screen. MODE 80,L shifts the display two characters to the left, and MODE 80,R shifts to the right. A third parameter, T, displays a test pattern on the screen for precise alignment. For example, MODE 80,R,T shifts the display to the right and prints the digits 0–9 eight times across the screen. Then you're asked if you can see the leftmost 0. If you can't, press N and the display shifts again followed by the same prompt. Press Y to return to the DOS prompt.

You can make the computer automatically switch to 80 columns when it's turned on by using a batch file. To create the batch file, insert one of your disks with system files on it and enter the command COPY CON AUTOEXEC.BAT. Then type MODE 80 and press the F6 function key (Fn-6) followed by Enter. When the drive stops spinning, display the directory: You should see the file AUTOEXEC.BAT. This file is automatically executed when you turn on the computer. For more information about batch files, see "All About IBM Batch Files" in the September and October 1985 issues of COMPUTE!. *Since MODE is an external DOS command, you must also have the file MODE.COM on the same disk. To add this feature to all your boot disks, use the COPY command to copy both AUTOEXEC.BAT and MODE .COM to each disk.* ©

COMPUTE! Subscriber Services

Please help us serve you better. If you need to contact us for any of the reasons listed below, write to us at:

COMPUTE! Magazine
P.O. Box 10954
Des Moines, IA 50340

or call the Toll Free number listed below.

Change Of Address. Please allow us 6–8 weeks to effect the change; send your current mailing label along with your new address.

Renewal. Should you wish to renew your **COMPUTE!** subscription before we remind you to, send your current mailing label with payment or charge number or call the Toll Free number listed below.

New Subscription. A one year (12 month) US subscription to **COMPUTE!** is $24.00 (2 years, $45.00, 3 years, $65.00. For subscription rates outside the US, see staff page). Send us your name and address or call the Toll Free number listed below.

Delivery Problems. If you receive duplicate issues of **COMPUTE!**, If you experience late delivery or if you have problems with your subscription, please call the Toll Free number listed below.

COMPUTE!
1-800-247-5470
In IA 1-800-532-1272

Help "Design" the On-Line Service of the Future!

Answering these questions can help you get *more* out of your PC...and determine if you qualify for a valuable free offer.

- Take this opportunity to try the new service at no charge in exchange for your feedback on the survey.
- Join scores of PC owners in this exciting developmental opportunity. It's like being there for the design of the PC or some new software.

1. What make and model of PC do you own?
☐ Commodore 64 or 128 ☐ Apple II
☐ IBM PC or Compatible ☐ Other _____ (make/model)

2. Do you own a disk drive? ☐ Yes ☐ No

3. Do you own a modem? ☐ Yes ☐ No
☐ No, but plan to buy within 6 months

4. Have you ever tried an on-line service? ☐ Yes ☐ No
If yes, would you consider using one? ☐ Yes ☐ No
If no, would you consider trying one? ☐ Yes ☐ No

5. Which of these credit cards do you use?
☐ VISA ☐ MasterCard
☐ American Express ☐ None

6. Please rate the following on-line services and features according to your own personal needs. (Use a 1 to 5 point scale, where "5" means "extremely important" and "1" means "not important at all.")

Services	Rate 1-5	Features	Rate 1-5
Current news/sports		Low cost—pennies/minute	
Banking services		No monthly subscription fees or minimums	
Encyclopedia		Local phone access	
On-line shopping		Fast and easy to learn, understand and use	
Advice from computer experts		English language command structure	
Airline reservations		Billable to credit card	
On-line communications with other PC owners			

THANK YOU. Please print the following information, so we can contact you if you qualify for the free offer or if we have any questions.

Your Name _____

Address _____ Apt No _____

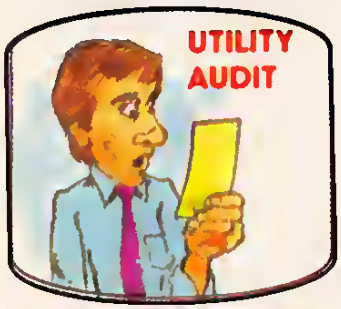
City/State/Zip _____

( ) Phone _____

See if you qualify for a free offer on this new service. Complete and mail survey today to... On-Line Service Survey. Attention: Steve Elliott, 625 N. Michigan Ave., Suite 1900, Dept. 8803, Chicago, Illinois 60611. CM-LB

FREE SOFTWARE

Now Get Up To 200 FREE Programs When You Subscribe to *COMPUTE!* Today



Subscribe to ***COMPUTE!*** today and you'll be getting a lot more than just another computer magazine. That's because ***COMPUTE!*** comes complete with up to 20 FREE programs in each big issue.

Subscribe now and you can depend on a steady supply of high quality, fun-filled programs like Cash Flow Manager, Speed Ski, Turtle Pilot, Boggler, Text Plot, Retirement Planner, and hundreds of other educational, home finance, and game programs the entire family can use all year long.

The free programs alone are worth much more than the low subscription price. But there's more to ***COMPUTE!*** than just free programs.

COMPUTE!'s superb articles deliver the latest inside word on everything from languages to interfaces...programming to disk drives. And our up-to-the-minute software reviews are must reading for any home user.

Whether you're a novice or an experienced user, ***COMPUTE!*** is perfect for you. So subscribe today. Return the enclosed card or call 1-800-247-5470 (in Iowa 1-800-532-1272).

COMPUTE! Publications, Inc. abc
One of the ABC Publishing Companies
825 7th Avenue 6th Floor New York NY 10019
Publishers of COMPUTE COMPUTE's Gazette COMPUTE's Gazette Disk COMPUTE Books and COMPUTE's Apple

**** FREE SOFTWARE ** FREE SOFTWARE ** FREE SOFTWARE ****

THE FUTURE OF MASS STORAGE

Selby Bateman,
Features Editor

Dramatic changes are occurring in the ways we store computer information. Technological advances and lower production costs are affecting both magnetic and optical data storage media. Traditional 5¼-inch floppy disks are giving way to 3½-inch microfloppies. Hard disk drives are rapidly becoming cost effective for average users. And low-power lasers are making optical storage technology the medium of the future. Here's a look at how far and how fast data storage technology has come, and where it's headed next.

Just when we think we're getting used to the pace of change, technology surprises us again. Consider the following important changes to the ways we store computer data:

• Apple Computer introduces its UniDisk 3.5, a 3½-inch disk drive for the Apple IIe and the Apple IIc computers that can store up to 800K (kilobytes) of information, more than five times the amount of the standard Apple 5¼-inch drives.

• New 3½-inch drives are included as standard storage systems for Atari's 520ST and Commodore's Amiga, joining Apple's Macintosh which was introduced with the drive in 1984. Industry sources believe IBM will also begin using the faster, more powerful 3½-inch drives sometime in 1986.

• Blue Chip Electronics says it plans to offer a 3½-inch disk drive for the Commodore 64, tentatively priced at about $100. Commodore insiders admit that they already have the technology to offer a 3½-inch drive and a 10-megabyte hard disk drive for the 64 and 128 (although no plans to market these peripherals have yet been announced). Atari also has been considering a 3½-inch disk drive for its line of eight-bit computers.

• Haba Systems is marketing a low-priced ($699) 10-megabyte (10,240-kilobyte) hard disk drive for the Atari ST. Prices for 10- and 20-megabyte hard disks fall as low as $400 for some computers. Hard disks on a card are announced for the IBM PC.

• Toshiba, Hitachi, Philips, and several other companies announce CD-ROM (Compact Disc-Read Only Memory) players that can store entire encyclopedias or massive software libraries on just a portion of a 4¾-inch optical laser disc.

• Maxell Corporation shows a new 2½-inch microfloppy disk drive that it plans to sell to manufacturers for use in laptop computers. The company also announces a 5¼-inch *eraseable, reusable* optical laser disc, which is to be marketed by 1987, and a new high-density *perpendicular* magnetic recording disk that packs up to 100K of data per inch.

Trusted Software and

Language Software

XREF-128 & XREF-64
BASIC cross-reference

Indispensible tool for BASIC programmers. Finds all references to variables, constants & line numbers. Sorts in alphabetical order. C-64 $17.95 C-128 $17.95

ASSEMBLER/MONITOR

Macro assembler and extended monitor. Supports all standard functions plus floating point constants. Monitor supports bank switching, quick trace, single step, more. $39.95

SUPER PASCAL

Full Pascal supports graphics, sprites, file management, more Supports pointers, dynamic memory management, machine language. Compiles to fast 6510 machine code. C-64 $59.95 C-128 $59.95

SUPER C COMPILER

Full compiler, Kernighan & Ritchie standard, but without bit fields. Includes powerful editor (41K source file); compiler, library (supports many functions) and linker. C-64 $79.95 C-128 $79.95

FORTH LANGUAGE

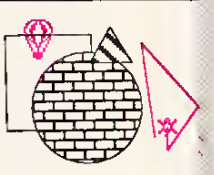
Based on Forth 79 (+ parts of '83). Supports hires graphics and sound synthesizer. Full screen editor, programming tools, assembler, samples, handbook. $39.95

MASTER

Professional development package for serious applications. Indexed file system, full screen & printer management, programmer's aid, multi-precision math, royalty-free runtime, more. $39.95

VIDEO BASIC

Add 50+ graphic, sound and utility commands to your programs with this super development package. Free distribution of RUNTIME version - no royalties! $39.95

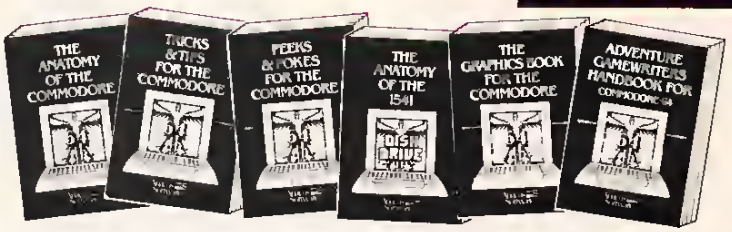


ADA TRAINING COURSE

Teaches you the language of the future. Comprehensive subset of language. Includes: editor; syntax checker, compiler; assembler; disassembler, handbook. $39.95

For Commodore Computers

Reference Books



ANATOMY OF C-64 Insider's guide to the '64 internals. Graphics, sound, I/O, kernal, memory maps, and much more. Complete commented ROM listings. 300pp $19.95

ANATOMY OF 1541 DRIVE Best handbook on this drive, explains all. Filled with many examples programs and utilities. Fully commented 1541 ROM listings. 500+pp $19.95

MACHINE LANGUAGE FOR C-64 Learn 6510 code & write fast programs. Many samples and listings for complete assembler, monitor and simulator. 200pp $14.95

GRAPHICS BOOK FOR C-64 Best reference, covers basic and advanced graphics. Sprites, Hires, Multicolor, 3D-graphics, IRQ, CAD, projections, curves, more. 350pp $19.95

TRICKS & TIPS FOR C-64 Collection of easy-to-use techniques: advanced graphics, improved data input, enhanced BASIC, CP/M, data handling and more. 275pp $19.95

1541 REPAIR & MAINTENANCE Handbook on the drive's hardware. Includes schematics & techniques to keep 1541 running. Align drive w/ & w/o scope. Large handbook size $19.95

ADVANCED MACHINE LANGUAGE Subjects not covered elsewhere: video controller, interrupts, timers, I/O, extensions to BASIC. Tips for the serious programmer. 210pp $14.95

PRINTER BOOK C-64/VIC-20 Understand Commodore, Epson compatible printers & 1520 plotter. Utilities, screen dump, 3D-plot, commented MPS-801 ROM listings. 330pp $19.95

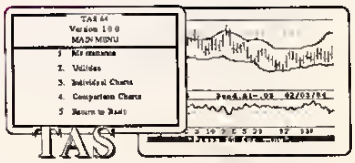
SCIENCE/ENGINEERING ON C-64 In-depth introduction to computers in science. Some topics covered are chemistry, physics, astronomy, electronics & others. 350pp $19.95

CASSETTE BOOK C-64/VIC-20 Make your cassette run faster than a disk drive! Cassette data-base, disk to tape backup, tape to disk, FastTape operating system. 225pp $14.95

Productivity Tools

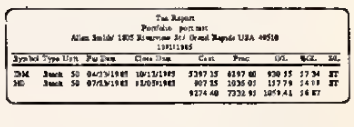
TECHNICAL ANALYSIS SYSTEM

A sophisticated charting and technical analysis system for serious investors. By charting and analyzing the past history of a stock, TAS can help pinpoint trends & patterns and predict a stock's future. TAS lets you enter trading data from the keyboard or directly from online financial services. $59.95



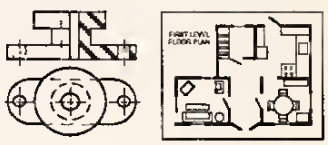
PERSONAL PORTFOLIO MANAGER

Complete portfolio management system for the individual or professional investor. Allows investors to easily manage their portfolios, obtain up-to-the minute quotes & news, and perform selected analysis $39.95



CADPAK

A deluxe graphics design and drawing package. Use with or without an optional lightpen to create highly-detailed designs. With dimensioning, scaling, text, rotation, object libraries, hardcopy. C-64 $39.95 C-128 $59.95



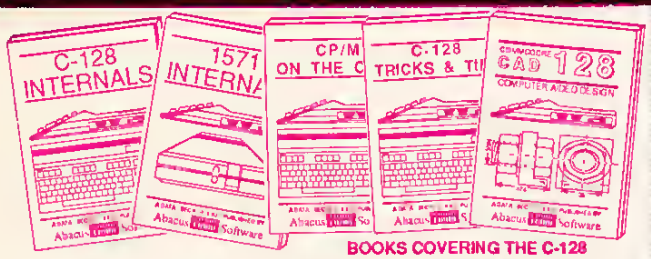
DATAMAT

Powerful, easy-to-use data management package using menu selections. Free-form design, 50 fields/record, 2000 records/disk. Sort on multiple fields in any combination. Complete selection and formatting for printing reports. $39.95

Authoritative Books

From Abacus Software

...a name you can count on



BOOKS COVERING THE C-128

IDEAS FOR USE ON C-64 Themes: auto expenses, calculator, recipe file, stock lists, diet planner, window advertising, others. Includes all program listings. 200pp $12.95

COMPILER BOOK C-64/C-128 All you need to know about compilers: how they work, creating your own and generating the final machine code. 300pp $19.95

Adventure Gamewriter's Handbook A step-by-step guide to designing and writing your own adventure games. Adventure game generator & four example games. 200pp $14.95

PEEKS & POKES FOR THE C-64 Includes in-depth explanations of PEEK, POKE, USR, and other BASIC commands. Learn the "inside" tricks about your '64. 200pp $14.95

OPTIONAL DISKETTES FOR BOOKS For your convenience, the programs contained in each of our books are available on diskette. All program thoroughly tested & error-free. Specify title of book when ordering. **$14.95 each**

C-128 INTERNALS Detailed guide presents the 128's operating system, explains the graphics chips, Memory Management Unit, and commented listing of Kernal. 500+pp $19.95

1571 INTERNALS Insiders' guide for novice and advanced users. Covers sequential & relative files, and direct access commands. Describes important DOS routines. Commented DOS listings. 500+pp $19.95

C-128 TRICKS & TIPS Chock full of info for everyone. Covers 80 column hi-res graphics, windowing, memory layout, Kernal routines, sprites and more. 300 pp $19.95

CP/M ON THE C-128 Essential guide to using CP/M on your 128. Simple explanations of the operating system, memory usage, CP/M utility programs, submit files and more. $19.95

COMPUTER AIDED DESIGN on your C-128 or 64. Create a CAD system using orograms provided. Covers 3D objects & rotation, MACROS, hatching, zooming, mirroring, line widths, dashed lines, more 300 pages. $19.95

Special Feature

BASIC 128

BASIC-128 is the *complete* compiler and development package for speeding up your BASIC programs.

BASIC-128 gives you many options: flexible memory management; choice of compiling in machine code, p-code or a mixture of both; use of a 40 or 80 column monitor; compiling in FAST-mode; etc.

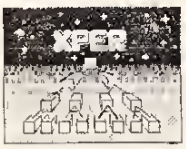
The extensive 80-page programmer's guide covers: all compiler options; error handling; array dimensioning; integer loops; interrupting compiled programs; BASIC extensions; memory usage; input/output handling; 80 column hi-resolution graphics.

BASIC-128 is the compiler for the programmer interested in optimizing the speed and preformance of their BASIC programs and protection of their invaluable programming techniques.

C-128 $59.95
C-64 $39.95

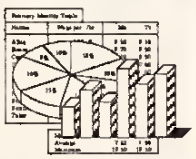
XPER

Capture your information on XPER's knowledge base and let this first *expert system* for Commodore computers help you make important decisions. Large capacity. Complete with editing & reporting. $59.95



POWERPLAN

One of the most powerful spreadsheets with integrated graphics for your Commodore computer. Includes menu or keyword selections, online help screens, field protection, windowing, trig functions and more. PowerGraph lets you create integrated graphs and charts from your spreadsheet data. $39.95

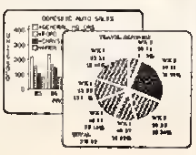


QUICKCOPY V2.0

Back up your valuable data with the fastest disk copier we've seen to date. Copies an entire disk in two and a half minutes on two drives or three and a half on one. $19.95

CHARTPAK

Make professional-quality pie, bar and line charts, and graphics from your data. Includes statistical functions. Accepts data from CalcResult and MultiPlan. C-128 has 3X the resolution of the C-64 version. Outputs to most printers. C-64 $39.95, C-128 $39.95



Ordering Information

Abacus Software

MasterCard VISA AMERICAN EXPRESS Card

P.O Box 7211 Grand Rapids, Michigan 49510

For Postage and handling include $4.00 per order. Foreign orders include $10.00 per item. Money order and checks in U.S. Dollars only. MasterCard, VISA and American Express accected. Michigan residents please include 4% sales tax.

For *fast* service call (616) 241-5510 Telex 709-101

For **free** catalog, please return this coupon or a copy to:
Abacus Software, P.O. Box 7211, Grand Rapids, MI 49510

PHONE: (616) 241-5510

Name_____
Address_____
City_____
State_____ Zip_____

3/86

• Sony announces a *writeable* optical laser disc storage system for computer use in business, science, and major archival applications capable of storing up to 3.2 *gigabytes* (3,276 megabytes, or 3,354,624 bytes) per disc.

Virtually every week, another advance in data storage technology surfaces within the computer industry. What's going to happen to all of the 5¼-inch floppy disks we're using now? Listen to Maxell's Ted Ozawa, vice president of the computer products division: "While we expect floppy disks to continue as a major industry factor for at least the next ten

Apple Computer's UniDisk 3.5 is a double-sided floppy disk drive that stores 800K of data, one of a growing number of 3½-inch drives for popular microcomputers.

years, new technologies offering more portability or more storage capacity are being developed more quickly than previously anticipated."

Ozawa's comments are being echoed throughout the computer industry as breakthroughs in storage technology are coupled with swiftly falling prices. Even casual computer users are beginning to think in terms of megabytes—and, with CD-ROMs, gigabytes.

The computer industry is rapidly advancing in two related areas of technology. The most immediate and visible changes are the advances in magnetic technology, ushering in low-cost, high-capacity disks and drives for the mass market. At the same time, a second technology is gaining speed, less visible but more important in the long run: laser discs designed for audio and video players are being tailored to computer data storage.

To understand the economies of scale involved with recent data storage improvements, consider that a typical 5¼-inch double-density IBM floppy disk holds approximately 360K of information. (By comparison, a Commodore 64 disk holds about 170K.) A double-sided 3½-inch disk contains approximately 800–880K of data. And an optical laser disc typically holds 550 megabytes, or the equivalent of almost 1,500 floppy disks (more than 3,500 Commodore 64 disks; more than 4,000 Apple II disks).

Such capacities are a far cry from the data storage devices used by many of the early microcomputer owners.A few years ago, modified audio cassette recorders were common storage devices on personal computers. They were inexpensive and usually reliable. Purchasers of Commodore VIC-20s, for example, and later Commodore 64 buyers, generally used Commodore Datassette recorders as a way to get started in computing for a fraction of the cost of a disk drive.

As with so much in the microcomputer field, magnetic tape storage was a descendant from mainframe computer systems. A cassette tape is a *sequential* access device. That is, tape moves sequentially across a recording head. In order to get to a program at the end of the tape, all of the preceding tape has to pass by the head first. The result is a frustratingly slow access time. More recent magnetic tape storage devices have used improved technology—data compaction, shorter loop tapes, and faster speeds—to remain competitive, at least as backup systems for hard disk drives.

With the advent of circular magnetic disks, also descendants of mainframe systems, many computer users decided to switch to the new medium. Although more expensive, *random* access storage offered significantly greater speed. A moveable read/write head could find information anywhere on the spinning disk almost instantaneously. The first floppy disks were either 8 inches (from IBM) or 5¼-inches (from Shugart) in diameter. But the emerging micro industry quickly agreed on the smaller 5¼-inch disks that predominate today.

During the past three years, an even smaller-sized magnetic disk, the 3½-inch format, has gained popularity. With its faster access speeds, 800K double-sided, double-density format, and sturdy plastic shell, the 3½-inch disk has definite advantages over the 5¼-inch standard. But when first introduced, so-called microfloppies came in at least three different sizes. Sony sold the 3½-inch disk, Dysan offered its 3¼-inch style, and Hitachi announced a 3-inch model. How did the 3½-inch disk become today's de facto standard?

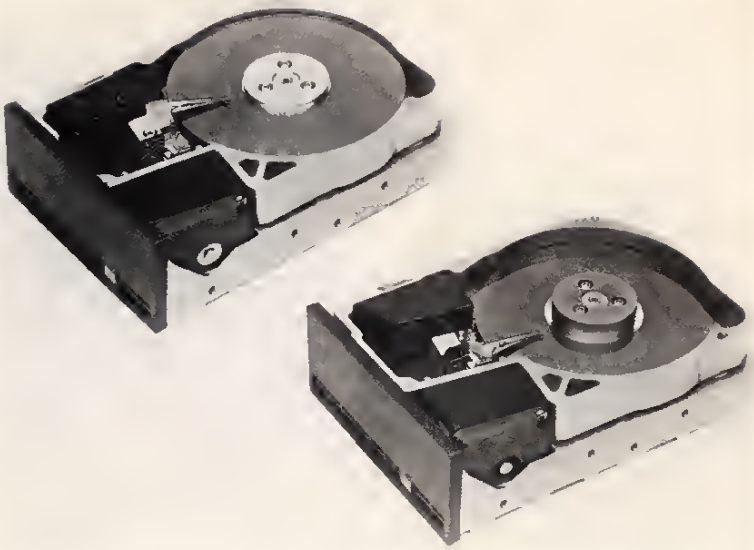
"The thing that happened was that Sony was very aggressive in promoting its format, not only to media [disk] makers but to drive makers," says David Berry, product manager for Maxell, a division of Hitachi. First Hewlett Packard and then Apple Computer adopted Sony's 3½-inch format, which created a snowball effect toward the Sony size. Hitachi still markets its 3-inch model, primarily in Japan

Maxell's new ultra-micro 2½-inch floppy disk holds 500K of unformatted data and is planned for use in laptop computers and other selected markets.

and Europe, notes Berry. In fact, Maxell just introduced an even smaller, 2½-inch disk, that it hopes to sell in selected market niches. "But," admits Berry, "we definitely think the 3½-inch will be the dominant force."

Regardless of their size, the physics of one floppy disk is similar to that of any other. An outer sleeve (vinyl for 5¼-inch; hard plastic for 3½-inch) protects a circular disk that rotates on a disk drive's spin-

Looking similar to a standard 3½-inch microfloppy, Maxell's new perpendicular recording disk stands magnetic particles on end to pack up to 100K of data per inch, about ten times the amount of a normal microfloppy.



Stripped-away views of two new 3½-inch Winchester-style 30- and 20-megabyte hard-disk drives from Peripheral Technology, Inc. Unlike floppies, these hard disks are nonremovable media.

ning hub at hundreds of revolutions per minute. There is a ferrous-oxide coating on one or both sides of the disk. The drive's read/write head (or heads, if the drive is double-sided) can read and alter the arrangement of magnetic particles .Information is recorded on the disk in concentric rings, or tracks, that are divided into arc-shaped sectors. Drive and disk manufacturers are continually improving this technology to allow increasing amounts of data to be accessed at faster speeds. The newest floppy disks are capable of megabytes of storage, such as the IBM AT's 1.2-megabyte floppy or Maxell's recently developed 10-megabyte metal-formula floppy disk, which contains 41.7K storage space per *track* and 120 tracks per side.

The next quantum leap in magnetic computer storage media is coming soon in the form of *perpendicular* recording technology. Magnetic floppy disks have heretofore used a standard metal oxide coating in which the particles lie horizontally on the disk surface. Perpendicular, or vertical, recording is analogous to the principle that more people can occupy a given space standing shoulder-to-shoulder than lying down side-by-side, explains Maxell's Ozawa.

Picture the magnetic particles "like a thickly clustered crowd of people standing in a field," he says. Maxell has developed a perpendicular high-density disk that allows 100K of data per inch, almost a tenfold storage increase over current recording densities. The company has worked with Hitachi to develop a metal-ferrite recording head that provides better head surface contact to read the densely packed particles. Other companies, chiefly Sony, Toshiba, and Matsushita, have issued technical papers and developed prototypes. But don't expect to see the perpendicular disk on store shelves for awhile. Perpendicular recording has been on the drawing boards for several years, but still hasn't proven to be as cost effective or as easily produced as traditional magnetic media, says Maxell's David Berry.

"There continues to be a lot of work by the media and drive people; however, the progress has been much slower than anticipated," he says. "It's a new technology, and the big thing today is the cost of storing per byte on any sort of media. It's a price-performance question right now as to whether it can be made cost effective. We feel that, down the road, it will be *the* media and drive of the future."

James Porter, head of the market research company, Disk/Trend, Inc., agrees that there's plenty of work ahead before perpendicular recording is durable and cost efficient enough to work.

On another front, computer users are finding that Winchester-style hard disk drives are increasing in performance as they drop in price. Lower prices and ease of use—especially important with today's increasingly integrated, memory-hungry applications—are making hard disks attractive even to casual computer users.

A hard disk spins within a drive, much like a floppy, but at faster speeds (3,600 rpm, for example). However, hard disks have traditionally been nonremoveable, and their recording heads don't actually touch the disk—instead, they float just above the surface. In the past, hard disks also cost thousands of dollars, were quite sensitive to dust and smoke, and were prey to "head crashes" that could ruin the whole disk.

Improvements in technology are now bringing prices down, sometimes well below a thousand dollars. In addition, new 3½-inch hard drives are being introduced along with the standard 8-inch and 5¼-inch models. These new systems are less prone to head crashes, have fewer problems with dust and smoke, and pack as much data into their systems as the older models.

Prices for hard disks in the 10- or 20-megabyte capacities range from $400 to $1,500 depending on access time, capacity, and other fea-

A new hybrid data storage device for IBM PCs and compatibles, the Clasix DataDrive Plus Series from Reference Technology combines a 550-megabyte CD-ROM optical disc player in the same box with a 10- or 20-megabyte Iomega Bernoulli Box removable magnetic cartridge.

tures. Some 300,000 of the 3½-inch hard drives were shipped in 1985, while about three million 5¼-inch hard drives (under 30 megabytes) shipped worldwide during the same period. The numbers for 3½-inch hard disks should increase appreciably during 1986, notes Porter.

Related to hard disk drives is a relatively new product, the Bernoulli Box from Iomega Corp. The Bernoulli Box actually *floats* its disk on a cushion of air within the drive, and also allows the disk to be removed—hence, it offers the portability of a floppy with the storage capacity of a hard disk. The floating disk also cuts down on the potential for destructive head crashes and the problems of dust and smoke.

Some computer experts believe that by the year 2000, the days of magnetic computer data storage may be only an historical footnote. Major advances in the use of low-power lasers in audio and video players are being quickly applied to computer technology. One of the hottest consumer electronics items in recent years is the audio compact disc (CD). And later this year, computer users will get a chance to see what CD laser technology can do when linked with a computer—virtually *any* computer—as a CD-ROM storage device.

The basic principle of CD ROMs is similar to the audio CD. A low-power laser beam reads microscopic pits that have been burned into the disc itself. These pits—representing a series of ones and zeros—contain the data that in a magnetic medium would be formed by the arrangement of magnetic particles. The 4.7-inch CD-ROM discs contain a whopping 550 megabytes of data per disc. The first applications are likely to be encyclopedias, such as the nine-million-word *Academic American Encyclopedia*, a 21-volume reference work that fits on just a quarter of one CD-ROM disc.

The biggest problem with CD-ROM technology at this point is that the devices are read-only. Unlike the magnetic particles on a floppy or hard disk, once the pits are burned into the surface of a CD, they can't be altered. But that limitation is already being challenged in the labs.

Sony recently announced a *writeable* 12-inch optical disc system that can hold up to 3.2 gigabytes of information. The disc is composed of two metallic elements sealed in a polycarbonate plastic. The laser beam writes information on the disc by turning the elements into an alloy which has different reflective properties. "This direct-seal method is more reliable and less costly than melt-type or bubble formation methods, which form gases during the writing process," says Robert Mesnik, Sony Information Products product manager. "The direct-seal method has a simple structure with no air spaces which can cause degradation of information over time."

This is a form of WORM (Write-Once, Read-Mainly) storage technology which offers high-density storage options for a variety of markets. The next step, however, is to create an optical technology that allows a laser to repeatedly write information on the same disc. Although not yet fully developed, an eraseable, reuseable 5¼-inch optical disc has been announced by Maxell for distribution in 1987. But for now, CD-ROMs will remain read-only reference and archival storage devices.

One of the first CD-ROM models debuting in 1986 is Toshiba's XM-1000 drive, which will be able to access digital computer data and also play music—that is, it will be both an audio accessory and a computer peripheral. The unit will have a storage capacity of 600–680 megabytes and may enter the retail market at close to $1,000. Sony will also market its CDU-1 CD-ROM player in 1986.

Five years ago, few computerists could have predicted how fast and how far data storage devices would come by the mid-1980s. Just as microcomputers themselves continue to grow in capability and diminish in price, so too will their storage devices expand to accommodate bigger memories, more complex integrated software, and as-yet unheard of applications. ©

With NRI training at home, you can . . .

Move up to a high paying career servicing computers

And you can start by actually building NRI's 16-bit IBM-compatible computer.

You can create your own bright, high paying future as an NRI trained computer service technician. The biggest growth in jobs between now and 1995, according to Department of Labor predictions, will occur in computer service and repair, where demand for trained technicians will double. There is still plenty of room for you to get in on the action—if you get the proper training now.

Total computer systems training, only from NRI

To learn how to work on computers, you have to get inside one. And only NRI takes you inside a computer, with total systems training that gives you hands-on experience with computers, peripherals, and software. As part of your training, you'll build a Sanyo MBC-550-2, which experts have hailed as the "most intriguing" of all the new IBM-compatibles.

Even if you've never had any previous training in electronics, you can succeed with NRI training. You'll start with the basics, rapidly building on the fundamentals of electronics until you master advanced concepts like digital logic, microprocessor design and computer memory. You'll probe into electronic circuits, using the exclusive NRI Discovery Lab® and professional Digital Multimeter, that you keep.

Learn to service today's computers

You'll assemble Sanyo's intelligent keyboard, install the power supply and disk drive, and attach the high resolution monitor—all the while performing hands-on experiments and demonstrations that reinforce your skills.

As you complete your Sanyo, you grasp the "secrets" that qualify you for a new career. You'll learn to program in BASIC and machine language. You'll use utility programs to check out the Sanyo 8088 microprocessor (the same chip used in the IBM PC). And you also get over $1,000 worth of software, including WordStar and CalcStar.

Learn the basics at home

Most importantly, you'll understand the principles common to all computers. Only a person who fully understands all the fundamentals can hope to be able to tackle all computers. NRI makes sure that you'll gain the knowledge and skills to maintain, troubleshoot and service computers.

With NRI training, you'll learn at home on your own time. That means your preparation for a new career or part-time job doesn't have to interfere with your current job. You'll learn at your own pace, in the comfort and convenience of your own home. No classroom pressures, no rigid night school schedules. You're always backed up by the NRI staff and your instructor, who will answer questions, give you guidance and be available for special help if you need it.

Send for free NRI catalog

Let others worry about computers taking their jobs. With NRI training, you'll soon have computers making good paying jobs for you. Send the coupon today for NRI's 100-page catalog, with all the facts about computer training. If the coupon is missing, write to NRI Schools, 3939 Wisconsin Ave., Washington, D.C. 20016.

IBM is a Registered Trademark of International Business Machines Corporation

NRI

McGraw-Hill Continuing Education Center
3939 Wisconsin Avenue, Washington, DC 20016
We'll give you tomorrow.

For Career courses approved under GI bill
☐ check for details

☐ **CHECK ONE FREE CATALOG ONLY**

- ☐ **Computer Electronics with Microcomputers**
- ☐ Data Communications
- ☐ Robotics & Industrial Controls
- ☐ Color TV, Audio, and Video System Servicing
- ☐ Electronic Design Technology
- ☐ Digital Electronics
- ☐ Communications Electronics
- ☐ Industrial Electronics
- ☐ Basic Electronics
- ☐ Telephone Servicing
- ☐ Small Engine Servicing
- ☐ Appliance Servicing
- ☐ Automotive Servicing
- ☐ Air Conditioning, Heating, Refrigeration, & Solar Technology
- ☐ Building Construction
- ☐ Locksmithing & Electronic Security

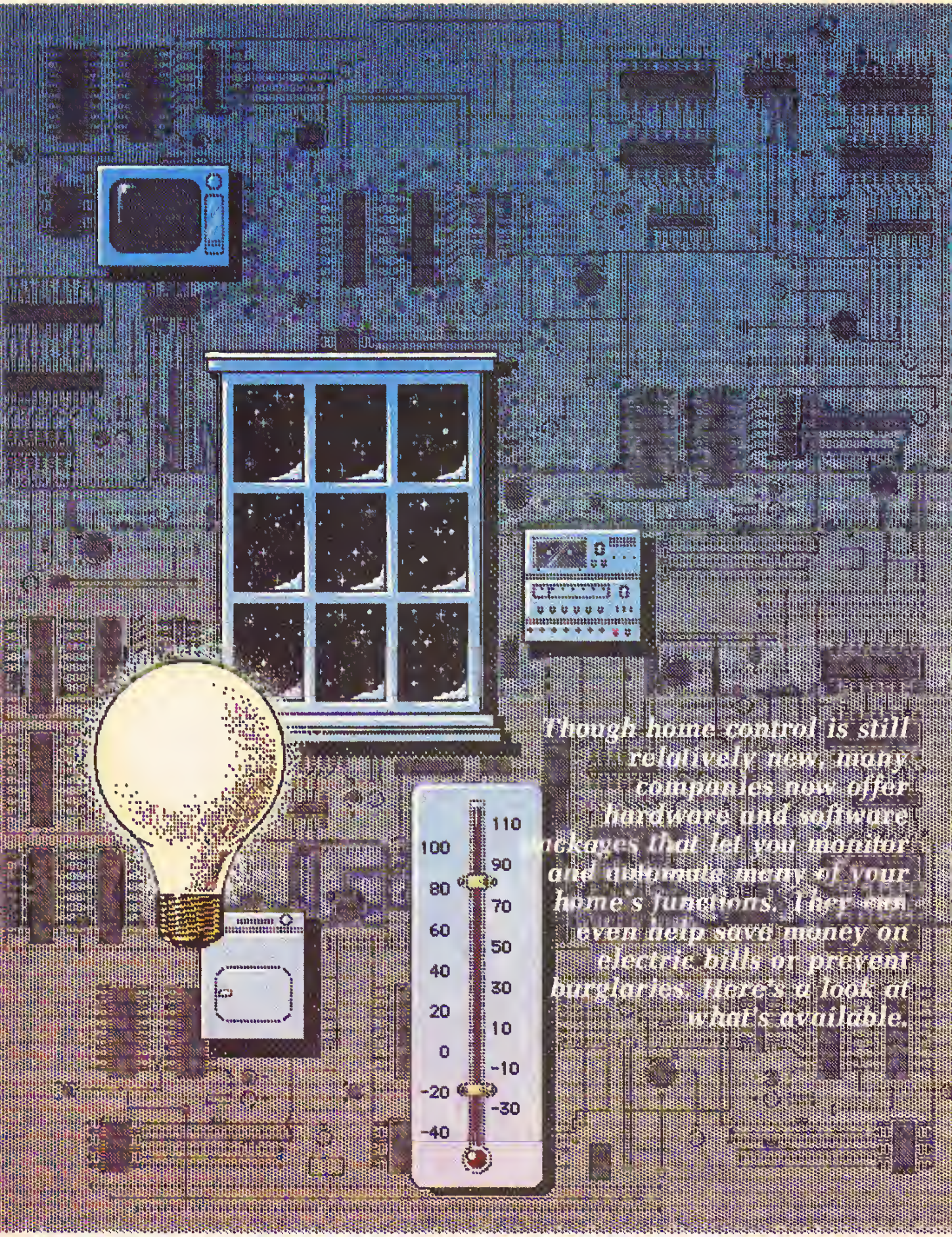
Name (Please Print) Age

Street

City/State/Zip

Accredited by the National Home Study Council

198-036



Though home control is still relatively new, many companies now offer hardware and software packages that let you monitor and automate many of your home's functions. They can even help save money on electric bills or prevent burglaries. Here's a look at what's available.

The COMPUTERIZED HOME

Kathy Yakal, Assistant Features Editor

Perhaps you already have some "smart" appliances in your home—a coffeepot that comes on automatically at a preset time every morning; a clothes dryer that senses what kind of fabric is being dried and acts accordingly; lights that turn on at dusk and off at dawn to discourage burglars when you're out of town; or a microwave oven that does everything but get in the car and drive to the grocery store.

The next step, it would seem, is to have all of these individual appliances controlled by a central unit, allowing you to talk to them through one keyboard, and which might even let them communicate with each other. The efforts of several manufacturers toward standardization is bringing this science-fiction scenario closer to home.

Many different home control units are now available. Some work through a personal computer and others are stand-alone units. They range in price from a few hundred dollars to a few thousand, and vary in the degree of technical expertise required to install them. Some are simple systems that work through existing household wiring via inexpensive plug-in modules. Others require some knowledge of programming and skill with a soldering iron, but are suitable for more sophisticated applications. Despite these different features, the three most common functions of these systems are appliance control, energy management, and home security.

Home control can be a full-time job for a computer, especially if you're using it to manage energy consumption. So unless you bought your computer specifically for home control, you're limited to buying a stand-alone unit or a second computer, either of which can be major investments.

Several home-control systems have been designed for the reasonably priced Commodore computers. You can still pick up a discontinued VIC-20 for under $100 at some stores, and a Commodore 64 for under $150. Dedicating such an inexpensive machine to one major function has proven very appealing to both manufacturers and consumers.

For instance, the X-10 Powerhouse is a very easy-to-use, inexpensive home control system that runs on the Commodore 64. The package consists of an interface box that plugs into the computer and software that runs the system. Up to eight different appliances can be set to turn on or off at specified times. The appliances must be plugged into modules available directly from X-10 USA for $16.95 each, or similar modules found at many electronics or hardware stores. (BSR modules are the most common.)

No programming knowledge is necessary to use the X-10's software. The opening screen shows nine icons representing different rooms in a house. After choosing a room, you "install" your own icons to show where appliances in your own house are. Then you simply set up a schedule for turning things off and on. The only time the system ties up the computer is when you're initially setting up or changing the schedule, so you can continue to use your computer as you normally would.

The X-10 Powerhouse is also available for the Apple IIe and IIc, Macintosh, and IBM PC series. The Macintosh version lets you draw your own house plans with *MacDraw* and *MacPaint* instead of using the boilerplate menu. All versions retail for $150. And if you want to do more than simply control appliances, additional modules and controllers include a burglar alarm interface, a thermostat setback controller, a telephone responder (which lets you control your home from any phone), and a heavy-duty 220-volt appliance module.

Many of the modules that work with the X-10 are also compatible with a home control system called HomeMinder, from General Electric. The HomeMinder software also dis-

plays graphic icons to help you set up the system. But unlike the X-10, the HomeMinder is a stand-alone unit that plugs into any color TV. Suggested retail price is $499. GE also sells a 25-inch color TV with the home control unit built-in for $1,200.

Savergy, Inc., markets two home control units compatible with the Commodore 64 and VIC-20. The PowerPort, which sells for $99.95, can control up to eight small appliances. The CIM 112, selling for $479, can handle larger appliances such as washing machines and water heaters.

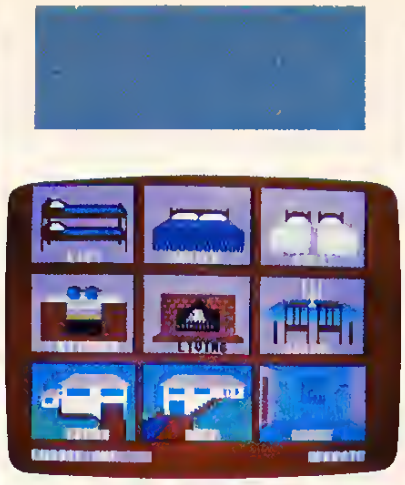
Genesis Computer Corporation also has a line of inexpensive home control units for the Commodore 64 and VIC-20. The VIController ($69.95) uses BSR-type modules to automate lights and small appliances. COMsense ($69.95), used in tandem with the VIController, lets you set up a home security system by hooking up the computer to switches on doors and windows. It can also be programmed to sense things like temperature or moisture levels in the air or ground, which would signal the VIController to turn on the lawn sprinkler or turn off the heat.

The COMclock ($69.95) is a battery backup for the system. It automatically reboots the software used by the VIController if there is a power interruption. Super Schedule Plus is a software package that integrates the operation of all three products. (COMsense and COMclock are compatible with Savergy's products.)

The Energy Manager, from Powerline Software, does not actually control appliances, but is a software package that helps keep track of and analyze energy use in homes and small office buildings. It runs on the Commodore 64 and retails for $59.95.

To justify the expense of adding a controller to your collection of home electronics, there has to be some reward. If it's primarily a home security system, or even just a unit that turns lights on and off, the rewards are obvious: security and convenience.

Another tangible reward can come in the form of monetary savings, if you buy a system geared



toward economizing your energy usage. John Helwig, of Jance Associates, Inc., has designed an inexpensive, easy-to-install system that can save substantial amounts of energy, especially if you have an all-electric home.

"The utility companies are in a bind," says Helwig. "They sell electric, and the electric they sell comes from the plants they build. Of course, they want to get away from the idea of building more plants. They want to sell you more electric, but they want to sell it to you at off-peak hours, because it is actually cheaper for them.

"During the day, when everyone wants power, they have to run their most inefficient plants. They only run their most efficient plants at night. So although they want to sell more power—like any company, they want to sell more of their product—they want to sell it in off-peak hours. So the kick in it for them is, very simply, if they can get their customers to use electric at night, it's to their advantage."

Helwig's energy management system, *REDUCE* (Reduction of Electrical Demand Using Computer Equipment), takes advantage of the time-of-day rates offered by many utility companies. (By lowering the hourly rates in evenings and on weekends, time-of-day rates encourage people to limit their heaviest electric use to off-peak hours.) *REDUCE* shuts things down during peak hours to make the most of these lower rates. In Helwig's own home, the system cut his electric bill by 40 percent last year.

Helwig's system so impressed Pennsylvania Power & Light that the utility company is test-marketing it with a group of consumers who have a wide variety of house sizes and lifestyles (and no computer background). The testing is designed to see how much money *REDUCE* can save, and whether it creates any inconveniences.

Compatible with the Commodore 64, *REDUCE* costs $250. In addition, Jance offers a Security Control System—a home security system with some home control features—for $195 (wired version) and $349 (wireless). It works on the 64 and VIC-20.

All home control systems have one thing in common. They must take analog information—anything read according to a scale, like degrees, volts, and pounds—and convert it into the digital information that computers can understand.

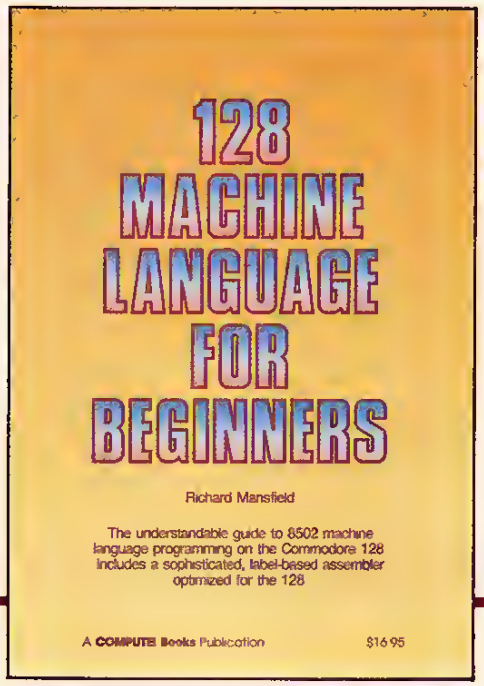
One such converter is the ADC-1 Data Acquisition and Control System ($449), from Remote Measurement Systems. It can be used with any RS-232-compatible computer, including the Commodore 64.

One application for an analog-to-digital converter is thermostat control. If the temperature outside drops considerably, a house takes longer to warm up. The ADC-1, using a smart thermostat program, wouldn't let it cool down as much, so it costs less to reheat it. Or if the ADC-1 is connected to security sensors and it detects a back window vibrating, it might wait to see if the window vibrates again. If it senses another vibration, or if the window breaks, the unit might turn on a sequence of room lights to make it appear that someone is walking in that direction to investigate.

The ADC-1 is currently being used for a tremendous variety of purposes in homes and business across the country. Amana, a major appliance manufacturer, has reduced the time required to test room air conditioners from 20 minutes to 2, using an ADC-1 and Commodore 64. A southern California architect uses the system to manage the components of a custom-designed solar heating system. And an arboretum on Bainbridge Island, Washington, uses it to keep track of meteorological measurements.

TAP THE POWER of the Commodore 128

By the author of *Machine Language for Beginners* and *Second Book of Machine Language*



128 Machine Language for Beginners

Richard Mansfield

One of the bestselling computer books ever has now been completely revised for the Commodore 128. Most commercial software is written in machine language because it's far faster and more versatile than BASIC. This new edition of *Machine Language for Beginners* is a step-by-step introduction to 8502 machine language programming on Commodore's 128 computer.

The book includes everything you need to learn to effectively program the 128: numerous programming examples, memory management tutorials; a complete description of the many Kernal routines and other new 128 features; numerous hints and programming techniques; and a dictionary of all major BASIC commands and their machine language equivalents. It also includes a high-speed, professional-quality, label-based assembler, optimized to take advantage of the speed and extra memory of the 128.

0-87455-033-5

$16.95

Like the other top-quality books from COMPUTE!, *128 Machine Language for Beginners* brings you ready-to-use information in a clear, lively style that makes learning easy and enjoyable, whether you are a beginner or an advanced computer user.

An optional disk is also available which includes the assembler and example programs in the book. The *128 LADS Disk* is fully tested and ready to load on the Commodore 128. It costs only $12.95 and saves you hours of typing time.

Order your copy of *128 Machine Language for Beginners* and the *LADS Disk* today. Call toll free 1-800-346-6767 (in NY 1-212-887-8525) or mail your payment (plus $2.00 shipping per book or disk) to COMPUTE! Books, P.O. Box 5038, F.D.R. Station, New York, NY 10150.

COMPUTE! Publications, Inc.
One of the ABC Publishing Companies
825 7th Avenue 6th Floor New York NY 10019
Publishers of COMPUTE! COMPUTE!'s Gazette COMPUTE!'s Gazette Disk COMPUTE! Books and COMPUTE!'s Apple Applications

COMPUTE! books are available in the U.K., Europe, the Middle East, and Africa from Holt Saunders, Ltd., 1 St. Anne's Road, Eastbourne, East Sussex BN21 3UN, England and in Canada from Holt, Rinehart, & Winston, 55 Horner Avenue, Toronto, ON M8Z 4X6.

COMPUTER MAIL ORDER

THE CMO ADVANTAGE

- THE BEST PRICES!
- Next day shipping on all in stock items
- Free easy access order inquiry
- Orders from outside Pennsylvania save state sales tax
- Free technical support with our factory trained technical staff
- There is no limit and no deposit on C.O.D. orders
- There's no extra charge for using your MasterCard or Visa. Your card is not charged until we ship.
- No waiting period for cashiers checks
- We accept purchase orders from qualified corporations. Subject to approval.
- Educational discounts available to qualified institutions
- FREE CATALOG MEMBERSHIP

ORDER LINE

CALL TOLL-FREE
1-800-233-8950
Educational Institutions Call Toll-Free
1-800-221-4283
CUSTOMER SERVICE & TECH SUPPORT
1-717-327-1450
Dept. A203

MAILING ADDRESS

Computer Mail Order
Dept. A203
477 East Third Street
Williamsport, PA 17701

MEMBER DIRECT MARKETING ASSOCIATION

CREDIT CARDS

VISA MasterCard AMERICAN EXPRESS DINERS CLUB

SHIPPING

Add 3%, minimum $7.00 shipping and handling on all orders. Larger shipments may require additional charges.

All items subject to availability and price change.

Returned shipments may be subject to a restocking fee.

CANADIAN ORDERS

1-800-268-3974
Ontario/Quebec
1-800-268-4559
Other Provinces
1-416-828-0866
In Toronto
TELEX: 06-218960

2505 Dunwin Drive,
Mississauga, Ontario
Canada L5L1T1

All prices shown are for U.S.A. orders.
Call The Canadian Office for Canadian prices.

HOME COMPUTERS

ATARI

130XE (128K)	CALL
520ST (512K)	CALL
800XL 64K	CALL
1010 Recorder	$49.99
1050 Disk Drive	CALL
1020 Printer	$29.99
1027 Letter Quality Printer	$129.00
1030 Direct Connect Modem	$59.99

Software Specials

8036 Atari Writer	$24.99
Star Raiders	$4.99
Missile Command	$4.99
Defender	$4.99
Galaxian	$4.99
Asteroids	$4.99
Centipede	$4.99
Miner 2049er	$4.99
Eastern Front	$4.99
VisiCalc	$39.99
Arcade Champ	$9.99

ATARI 520-ST SOFTWARE

SIERRA ON LINE

Ultima II	$39.99
Gato	$29.99
King's Quest	$37.99

BATTERIES INCLUDED

D.E.G.A.S.	$29.99

INFOCOM

Zork I, II, III	(ea.) $29.99
Hitchiker's Guide	$29.99
Wishbringer	$29.99
Suspended	$37.99

HABA

Hippo-C	$44.99
Haba Writer	$44.99

MIRAGE CONCEPTS

Express	$34.99

MARK OF THE UNICORN

Final Word	$94.99
Hex	$29.99
PC Intercom	$89.99

V.I.P.

Professional	CALL

APPLE

APPLE IIe	CALL
APPLE IIc	CALL
MacINTOSH	CALL
IIc LCD Display	CALL

HAYDEN

Art Grabber	$31.99
Home Design	$49.99
Media Works	$63.99

PALADIN

Crunch 512	$189.00

Macintosh Software

Lotus Jazz	CALL
Microsoft Excel	$259.00
Living Videotext	
ThinkTank 512	$159.00
Manhetten Ready, Set, Go	$79.99
Creighton Development	
Mac Spell	$69.99
Monogram Dollars & Sense	$99.99
Peachtree Back to Basics - GL	$109.00
PFS File & Report (New Version)	$119.00
Silicon Beech Airborn	$25.99

commodore

C128 Computer	$269.00
C1571 (Disk Drive for C128)	$NEW
C1902 (RGB 13" Monitor for C128)	$NEW
C1670 (Modem for C128)	$NEW
CBM 64	CALL
C1541 Disk Drive	$199.00
C1530 Datasette	$39.99
M-801 Dot Matrix Printer	$159.00
MCS 803 Dot Matrix	$179.00
C1802 Color Monitor	$189.00
C1660 Auto Modem	$59.99
DPS 1101 Daisy Printer	$339.00

Professional Software

Fleet System II w/Spell (128)	$49.99
Trivia Fever	$29.99
Word Pro 4 Plus/5 Plus each	$239.00
Info Pro	$179.00

BRODERBUND

The Print Shop	$29.99
Music Shop	$29.99

pfs:

File (64)	$39.99

BATTERIES INCLUDED

PaperClip w/Spell Pack	$49.99
The Consultant DBMS	$37.99
Bus Card II	$119.00
80 Col Display	$99.99

MODEMS

ANCHOR

Volksmodem	$59.99
Volksmodem 300/1200	$189.99
Signalman Express	$259.00
Lightning 2400 Baud	$399.00
Expressi	$199.00

OIGITAL DEVICES

AT300 - 300 Baud (Atari)	$99.99

Hayes

Smartmodem 300	$139.00
Smartmodem 1200	$389.00
Smartmodem 1200B	$359.00
Smartmodem 2400	$599.00
Micromodem IIe	$149.00
Smart Com II	$89.99
Chronograph	$199.00
Transet 1000	$309.00

AST

Reach 1200 Baud Half Card	$399.00

MICROBITS

MPP-1064 AD/AA (C-64)	$69.99

Novation

Smart Cat Plus	$299.00
J-Cat	$99.99
Novation 2400	$549.00
Apple Cat II	$229.00
212 Apple Cat II	$379.00
Apple Cat 212 Upgrade	$229.00
Macmodem	$279.00

QUADRAM

Quadmodem II	
300/1200	$339.00
300/1200/2400	$499.00

TELELEARNING

C64 300 Baud (Closeout)	$39.99

EVEREX

1200 Baud Internal (IBM/PC)	$199.00

GRAPHICS

Polaroid

Palette	$1299.00

PORTABLE COMPUTERS

HEWLETT PACKARD

41CV	$189.99
41CX	$249.99
HP 11C	$62.99
HP 12C	$89.99
HP 15C	$89.99
HP 16C	$89.99
HPIL Module	$98.99
HPIL Cassette or Printer	$359.99
Card Reader	$143.99
Extended Function Module	$63.99
Time Module	$63.99

We stock the full line of HP calculator products

NEC

PC-8401 LS	CALL
PC-8201 Portable Computer	$319.00
PC-8231 Disk Drive	$599.00
PC-8221A Thermal Printers	$149.00
PC-8281A Data Recorder	$99.99
PC-8201-06 8K RAM Chips	$79.99

SHARP

PC-1350	$149.00
PC-1261	$149.00
PC-1500A	$169.00
PC-1250A	$89.99
CE-125 Printer/Cassette	$129.00
CE-150 Color Printer Cassette	$149.00
CE-161 16K RAM	$129.00

DISKETTES

maxell

3½" SS/DD (10)	$24.99
3½" DS/DD (10)	$34.99
5¼" MD-1 w/Hardcases (10)	$12.99
5¼ MD-2 w/Hardcases (10)	$19.99
5¼" MD-2-HD for AT (10)	$39.99
3½" 5 pack SS/DD	$15.99

Verbatim.

5¼" SS/DD	$19.99
5¼" DS/DD	$24.99
Disk Analyzer	$24.99

Dennison

Elephant 3½ SS/DD	$29.99
Elephant 5¼" SS/SD	$13.99
Elephant 5¼ SS/DD	$14.99
Elephant 5¼" DS/DD	$16.99
Elephant Premium DS/DD	$22.99

IBM

5¼" DS/DD floppy disks (Box of 10)	$26.99

DISK HOLDERS

INNDVATIVE CONCEPTS

Flip-in-File 10	$2.99
Flip-in File 50	$17.99
Flip-in-File 50 w/lock	$24.99
Flip-in-File 100	$24.99

AMARAY

50 Disk Tub 5¼	$9.99
30 Disk Tub 3½"	$9.99

DRIVES

HARD

I·MEGA

10 meg Bernoulli Box	$1899.00
20 meg Bernoulli Box	$2599.00
5 meg "MacNoulli"	$1499.00

TALLGRASS TECHNOLOGIES

25 35 50 80 meg (PC)
from $1299.00

IRWIN

Tape Backup	CALL

EVEREX

60 Meg Internal Backup System	$799.00

U-SCI

10 meg Internal IBM	$399.00
20 meg Internal IBM	$549.00

CDRE

AT20-AT72MB	CALL

FLOPPY

INDUS

Atari GT	$219.00
C-64 GT	$219.00

1571

SD1 C 64 Single	$219.00
SD2 C 64 Dual	$469.00

Tandon

320K 5¼" (PC)	$119.00

TEAC

320K 5¼"	$119.00

CALL TOLL-FREE

COMPUTER MAIL ORDER

MONITORS

AMDEK

Video 300 Green	$129.00
Video 300A Amber	$139.00
Video 310A Amber TTL	$169.00
Color 300 Composite	$169.00
Color 500 Composite/RGB	$289.00
Color 600 Hi-Res RGB	$399.00
Color 710 Ultra Hi-Res	$439.00
Color 722 Dual Mode	$529.00

NEC

JB1270G/1275A (ea)	$99.99
JB1280G TTL	$129.00
JB1285G TTL	$129.00
JC1460 RGB	$249.00
JC1225 Composite	$179.00

PRINCETON

MAX-12E Amber	$179.00
HX-9 9" RGB	$469.00
HX-9E Enhanced	$519.00
HX-12 12" RGB	$469.00
HX-12E Enhanced	$559.00
SR-12 Hi-Res	$599.00
SR-12P Professional	$699.00

TAXAN

115 12" Green	$119.00
116 12" Amber	$129.00
121 TTL Green	$139.00
122 TTL Amber	$149.00
610 510x200 RGB	$NEW
620 640x200 RGB	$NEW
630 640x200 RGB	$NEW
640 720x400 RGB	$NEW

QUADRAM

8400 Quadchrome I	$499.00
8410 Quadchrome II	$339.00
8420 Amberchrome	$179.00
8500 Quad Screen	$1449.00

ZENITH

ZVM 1220/1230 (ea)	$99.99
ZVM 1240 IBM Amber	$149.00
ZVM 130 Color	$269.00
ZVM 131 Color	$249.00
ZVM 133 RGB	$429.00
ZVM 135 RGB/Color	$459.00
ZVM 136 RGB/Color	$599.00

INTERFACES

AST

Multi I/Q (Apple II)	$159.00

PRACTICAL PERIPHERALS

Graphcard	$79.99
Serial Card	$99.99
Microbuffer II+	$169.00
Microbuffer 32K	$189.00

QUADRAM

Microfazer	from $139.00
Efazer (Epson)	from $79.99

Orange Micro

Grappler CD (C64)	$89.99
Grappler + (Apple)	$89.99
Grappler 16K+ (Apple)	$159.00

DIGITAL DEVICES

Ape Face (Atari)	$49.99
U-Print A (Atari)	$54.99
U-A16/Buffer (Atari)	$74.99
U-Call Interface (Atari)	$39.99
U-Print C (C64)	$49.99
P-16 Print Buffer	$74.99
U-Print 16 apple IIc	$89.99

PRINTERS

Canon

A40	CALL
LBP-8A1 Laser	CALL

CITIZEN

MSP-10 (80 col)	$279.00
MSP-15 (132 col)	$389.00
MSP-20 (80 col)	$349.00
MSP-25 (132 col)	$509.00

C.ITOH

Prowriter 7500	$179.00
Prowriter 1550P	$349.00
Starwriter 10-30	$399.00

corona

Lazer LP-300	$2799.00

DIABLO

D25 Daisywheel	$549.00
635 Daisywheel	$899.00
D80IF Daisywheel	CALL

daisywriter

2000	$749.00

EPSON

Homewriter 10, LX-80, LX-90	CALL
FX-85, FX-286, RX-100, JX-80	CALL
DX-10, DX-20, DX-35	CALL
SQ-2000, Hi-80, HS-80, AP-80	CALL
LQ-800, LQ-1000, LQ-1500	CALL

JUKI

6000 Letter Quality	CALL
6100 Letter Quality	CALL
6200 Letter Quality	CALL
6300 Letter Quality	CALL
5510 Dot Matrix	CALL

LEGEND

808 Dot Matrix 100 cps	$179.00
1080 Dot Matrix 100 cps	$259.00
1380 Dot Matrix 130 cps	$289.00
1385 Dot Matrix 165 cps	$339.00

NEC

8027 Transportable	$199.00
3000 Series	$1099.00
8000 Series	$1399.00
ELF 360	$449.00
Pinwriter 560	$999.00

OKIDATA

182, 183, 192, 193, 2410, 84	CALL
Okimate 10 (Specify C64/Atari)	$189.00
Okimate 20 (IBM)	CALL

Panasonic

KX1080	$NEW
KX1091	$259.00
KX1092	$389.00
KX1093	$479.00

QUADRAM

Quadjet	$399.00
Quad Laser	CALL

SILVER-REED

500 Letter Quality	$279.00
550 Letter Quality	$419.00
770 Letter Quality	$759.00

star

SG-10C (C64 Interface)	CALL
SB/SD/SG/SR Series	CALL
Powertype Letter Quality	CALL

Texas Instruments

TI850	$529.00
TI855	$639.00
TI865	$799.00

TOSHIBA

1340 (80 column)	$469.00
P341 (132 column)	$949.00
P351 (132 column)	$1099.00

PC COMPATIBLES

IBM PC SYSTEMS

Configured to your specification. Call for Best Price!

IBM-PC, IBM-PC II, IBM-XT, IBM-AT

KAYPRO

KP-2000 Portable	CALL
Kaypro PC	CALL

SOFTWARE FOR IBM

ASHTON-TATE

Framework II	$399.00
dBase III	$369.00

BORLAND

Lightening	$59.99
Sidekick (unprotected)	$59.99
Reflex	$59.99

CENTRAL POINT

Copy II PC Backup	$29.99

DECISION RESOURCES

Chartmaster	$229.00
Signmaster	$169.00
Diagram Master	$219.00

ENERTRONICS

Energraphics/Plot	$289.00

FOX & DELLER

Quickcode III	$169.00

FUNK SOFTWARE

Sideways	$39.99

HARVARD SOFTWARE

Total Project Manager	$269.00

INFOCOM

Cornerstone	$279.00

LIFETREE

Votkswriter Deluxe	$159.00

LIVING VIDEOTEXT

Think Tank	$109.00
Ready	$64.99

LOTUS

Symphony	CALL
1-2-3	CALL

MECA SOFTWARE

Managing Your Money 2.0	$99.99

MICROSTUF SOFTWARE

Crosstalk XVI	$89.99
Crosstalk Mark IV	$149.00
Remote	$89.99

MICRORIM SOFTWARE

R:Base 4000	$249.00
R:Base 5000	$389.00
Clout 2.0	$129.00

MICROPRO

WordStar 2000	$249.00
WordStar 2000 +	$299.00
WordStar Professional	$199.00
Easy	$99.99

MICROSOFT

Word	$229.00
Mouse	$139.00
Flight Simulator	$39.99
MultiPlan	$129.00

MULTIMATE

Advantage	$299.00
Multi Mate Word Proc	$249.00
On File	$94.99
Just Write	$94.99

NOUEMENON

Intuit	$69.99

NORTON

Norton Utilities 3.1	$59.99

ONE STEP

Golf's Best	$39.99

PEACNTREE SOFTWARE

Peachtext 5000	$199.00

PFS:IBM

First Success W/F/P	$199.00
File/Graph (ea)	$79.99
Report	$74.99
Write/Proof Combo	$79.99

PROFESSIONAL SOFTWARE

Wordplus-PC w/Boss	$249.00

TNE SOFTWARE GROUP

Enable	$259.00

SATELLITE SYSTEMS

Word Perfect 4.1	$219.00

SORCIM/IUS

Accounting AP/AR/GL/INV/QE (ea)	$299.00
SuperCalc III	$199.00
EasyWriter II System	$199.00
Super Project	$199.00

BPI SOFTWARE

Open Access	$379.00

SUBLOGIC

Jet	$39.99

5th GENERATION

Fast Back	$119.00

ZENITH

PC-138 Series	CALL
PC-148 Series	CALL
PC-158 Series	CALL
PC-160 Series	CALL
PC-171 Series	CALL
AT-200 Series	CALL

SANYO

MBC 550-2 Single Drive	$649.00
MBC 555-2 Dual Drive	$949.00
MBC 675 Portable	CALL
MBC775	$1699.00
MBC 880 Desktop	CALL

AT&T

Safari (7300)	CALL
6300	CALL

corona

PPC400 Dual Portable	$1289.00
PPCXT 10 meg Portable	$1969.00
PC40022 Dual Desktop	$1389.00
PC400-HD2 10 meg	$1989.00

ITT

ITT X-TRA	
256K, 2 Drive System	CALL
256K,10 meg Hard Drive System	CALL
XP5, 20 meg	CALL

SPERRY

Sperry-AT	as low as $1749.00
Sperry-IT	as low as $2699.00
Call for Specific Configuration!	
All Models	CALL

MULTIFUNCTION CARDS

AST

Six Pack Plus	$229.00
I/Q Plus II	$139.00
Advantage-AT	$399.00
Graph Pak/64K	$599.00
MonoGraph Plus	$399.00
Preview Mono	$299.00
PC Net Cards	$379.00
5251/11 On-line	$669.00
5251/12 Remote	$579.00

dca

IRMA 3270	$879.00
IRMA Print	$999.00
IRMA Smart Alec	$779.00

EVEREX

Edge Card	$259.00
Graphics Edge	$259.00
Magic Card II	$169.00

HERCULES

Graphics	$299.00
Color	$159.00

IDEAssociates

IDEA 5251	$589.00

MYLEX

The Chairman	$469.00

PARADISE

Color/Mono Card	$149.00
Modular Graphics Card	$259.00
Multi Display Card	$219.00
Five Pack C, S	$129.00

PERSYST

Bob Board	$359.00

TECMAR

Captain - 64	$199.00
Captain Jr 128K	$199.00
Graphics Master	$469.00

QUADRAM

Quadport-AT	$119.00
Liberty-AT (128K)	$349.00
The Gold Quadboard	$449.00
The Silver Quadboard	$239.00
Expanded Quadboard	$209.00
Liberty	$309.00
QuadSprint	$499.00
QuadLink	$399.00
QuadColor	$199.00
QuadJr Expansion Chassis	$419.00
Expansion Chassis Memory	$199.00
Chronagraph	$79.99
Parallel Interface Board	$64.99

INTEL

PCNC8087 5MHz	CALL FOR YOUR PC
PCNC8087-2 8 MHz	
PCNC80287 6 MHz	
1010 PC-Above Board	
1110 PS-Above Board	
2010 AT-Above Board	

All the exciting, entertaining, and educational games, applications, and utilities from COMPUTE! magazine are now available on disk

for your Commodore, Atari, Apple, or IBM personal computer.

The COMPUTE! Disk

A new *COMPUTE! Disk* is published every month, rotating among the four major machines covered by *COMPUTE!*: Commodore 64 and 128; Atari 400/800, XL, and XE; Apple II-series; and IBM PC, PCjr, and compatibles.

Every three months you can receive a disk with all the quality programs from the previous three issues of *COMPUTE!* that will run on your brand of computer.

Like the popular *COMPUTE!'s Gazette Disk*, the *COMPUTE! Disk* is ready-to-load and error-free. It saves you valuable hours of typing time and eliminates typing errors.

With a subscription, you will receive one disk every three months for a total of four disks a year—for only $39.95. That saves you $20 a year off the single-issue cost.

Or you can order individual issues of the *Disk* for $12.95 a disk plus $2.00 shipping and handling.

Remember to specify your type of computer when ordering the *COMPUTE! Disk*. You'll find more information about this month's *COMPUTE! Disk* in this issue. (Note: You'll need the corresponding issues of *COMPUTE!* magazine to use the *Disk* since the disk will have no documentation.)

For fastest service when ordering a subscription to the *COMPUTE! Disk*, call toll free 1-800-247-5470 (in Iowa 1-800-532-1272).

For more details or to order individual issues of the *COMPUTE! Disk*, call our Customer Service Department toll free at 1-800-346-6767 (in New York 212-887-8525).

Please allow 4–6 weeks after placing an order for your first disk to arrive.

COMPUTE! Publications, Inc. abc
One of the ABC Publishing Companies
825 7th Avenue 6th Floor New York NY 10019
Publishers of COMPUTE! COMPUTE!'s Gazette COMPUTE!'s Gazette Disk COMPUTE! Books and COMPUTE!'s Apple Applications

The low cost of Commodore home computers prompted designers at Proteus Electronics, Inc., to develop the Simple Interface Data Acquisition System (ADAC) for the Commodore 64, 128, and VIC-20. The system consists of an interface ($34.95) and an analog data acquisition conditioner ($64.95). The system can digitize up to 16 channels of analog signals, making it appropriate for functions such as heating, cooling, solar control, voltage measurements, robotics, and weather station monitoring. An Apple version should be available by the time you read this for about $150.

For a home control unit to appeal to many people who don't own personal computers, ease of installation and use is crucial. Manufacturers realize that, and continue to work toward that goal.

Voice recognition may be one method of operation that could appeal to people not enamored of keyboards. Magician Gus Searcy and West German programmer Franz Kavan have developed a home control system that uses voice recognition. Marketed by Mastervoice, the system is called Sidney, the Butler in a Box. Working through existing household wiring, Sidney can dim and brighten lights, answer the phone, act as a security guard, and turn household appliances off and on—all at the sound of its master's voice. A stand-alone unit, Sidney retails for $1,195.

It's been predicted that eventually we'll all have some kind of universal controller in our homes, a unit that ties together all of our electronic appliances, entertainment equipment, and telephones. Fortunately, we can get a taste of the future right now with the easy-to-use products already available.

For more information about products mentioned here, contact:

General Electric Consumer Electronics
Business Operations
Portsmouth, VA 23705

Genesis Computer Corporation
1444 Linden Street
P.O. Box 1143
Bethlehem, PA 18018

Jance Associates, Inc.
P.O. Box 234
East Texas, PA 18046

Mastervoice
10523 Humboldt Street
Los Alamitos, CA 90720

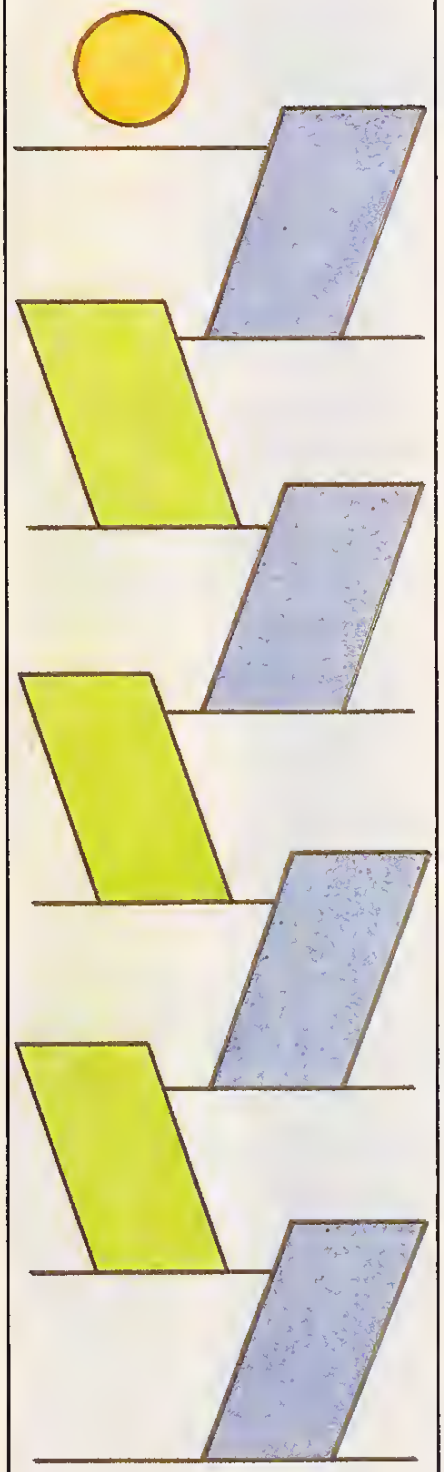
Powerline Software
P.O. Box 635
New Hartford, NY 13413

Proteus Electronics, Inc.
RD #2, Spayde Road
Bellville, OH 44813

Remote Measurement Systems
2633 Eastlake Avenue E.
Suite 206
Seattle, WA 98102

Savergy, Inc.
1404 Webster Avenue
Fort Collins, CO 80524

X-10 USA, Inc.
185A Legrand Avenue
Northvale, NJ 07647



Switchbox

Todd Heimarck, Assistant Editor

Here's a challenging game of strategy that looks easy at first, but takes time to master and permits many variations. The original program is written for the Commodore 128. We've added new versions for the Commodore 64, Apple II, eight-bit Atari computers, IBM PC/PCjr and Atari 520ST, as well as an Amiga version with speech and stereophonic sound effects.

Playing "Switchbox" is like putting dominos in place for a chain reaction—either you're setting them in position or you're knocking them over. Winning requires skill and a sense of when to go for points and when to lay back and wait for a better board. The goal is simple: You try to score more points than your opponent by dropping balls into a box full of two-way switches. Each switch has a trigger and a platform. If the ball lands on an empty platform, it stops dead. But if it hits a trigger, it reverses the switch and continues. In many cases dropping a single ball creates a cascading effect—one ball sets another in motion, which sets others in motion, etc., all the way down.

Type in the program listing for your computer, and save a copy before you run it. If you're using a Commodore 128, enter the 128 version in 128 mode (not 64 mode). The Atari version runs on any eight-bit Atari computer with at least 16K of memory. The Apple II game works on any Apple II machine, under either ProDOS or DOS 3.3. The IBM PC/PCjr version requires BASICA (PC) or cartridge BASIC (PCjr), and a color/graphics adapter on the PC.

The Amiga version of Switchbox requires 512K of memory and Amiga BASIC by Microsoft (not ABasiC, the version supplied with a few very early Amigas). If you have only ABasiC, contact your dealer about getting an upgrade to Amiga BASIC. Before running the program, make sure that the Amiga's display is set for 80 columns of text (if it isn't, the numbers printed on the screen won't match the rest of the display). If you've previously changed the display to 60 columns, open the Preferences icon and change it back to 80, then close Preferences, activate BASIC and run the program as usual.

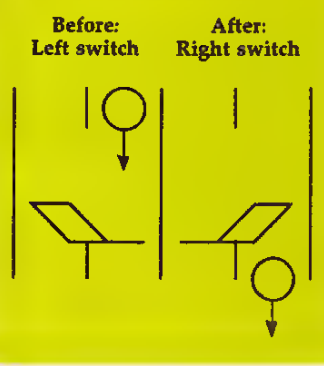
Since the Amiga program doesn't use line numbers, we've placed small arrows in the listing to show you where each line ends. Don't try to type in the arrows: There's no such character in the Amiga's character set. Wherever you see an arrow in the listing, press Return (or move the cursor off that program line) to enter the line. It's important to include the colon (:) that appears after program labels (*Nextround:*, *Setup:*, and so on). If you accidentally omit the colon, you'll make it impossible for the computer to use that label correctly.

Before typing in the 520ST version you must turn off the buffered graphics by clicking on Buf Graphics in the Run menu. You can be in either medium or low resolution mode when typing in the program; but before running it, you must be in low resolution mode. This can be done by selecting Set Preferences from the Options menu on the desktop and clicking on Low.

A Box Of Switches

Switchbox is a tale of twos: Each switch has two parts, two positions, two states, two paths in, and two paths out. The two parts are the platform and the trigger. A switch can lean to the left (platform left, trigger right) or to the right (platform right, trigger left):

Figure 1. Trigger States



The trigger is weak, and always allows balls to pass. But the platform is strong enough to hold a single ball. So the platform either holds a ball—it's full—or it does not and is empty. When a ball sits on a platform, the switch is said to be loaded, or full.

Figure 2. Loaded Trigger

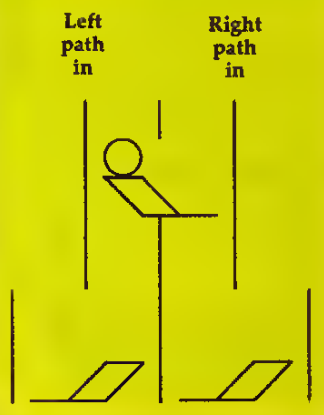


Figure 2 shows a full switch over two empty switches. The platform holds a ball and leans to the left. The trigger extends to the right. Note that the switch on top has two pathways leading in, the left path and the right, and that the right path leading out is the left path into one of the switches below. The left path of the top switch leads into the right path of the other, the switch below and to the left. If you drop a ball down the righthand path, it hits the trigger and flips that switch to the right. Then it continues down, hits the lefthand trigger below and flips that switch as well.

In the meantime, the ball on the platform is set in motion (when the switch is flipped) and then hits the trigger. The top switch is reset to point to the left. The second ball then drops a level to the platform below, where it stops. The playing field is composed of five levels, with four switches in the first level and eight in the bottom level. At the beginning of the game, there are no balls on the field—all platforms are empty—and the position of each switch is chosen randomly.

Moving Down The Path

Players alternate dropping balls into one of eight entry points. These balls (and others) may or may not make it all the way through the switchbox, to one of the 16 exit paths. Balls fall straight down (with one exception), so a ball's movement is always predictable. When it hits an empty switch, one of two things can happen. If it lands on the empty platform, it stops dead in its tracks. But if it lands on a trigger, it falls through to the next level below.

Moving balls always make it through loaded switches. Triggers allow balls to continue, and move the switch to the other position. If it's loaded, the dead ball on the platform is put into motion and it hits the trigger that just moved over. This makes the switch go back to its original position, but with an empty platform. So when a ball hits the trigger of a loaded switch, its motion continues unabated. The switch moves, the ball on the platform begins to fall and it hits the newly placed trigger. The newly emptied switch moves back again, and the two balls drop to the next level.

There's one more possibility: a ball dropping onto a platform that already holds a ball. A platform can't hold any more than one ball, so when this happens one of the balls slides over to the trigger. So the ball does not move straight down—it slides over to the next pathway. This is the exception to the rule that balls drop in a straight line. Of course, when the ball hits the trigger, the switch changes position, causing the other ball to drop and hit the trigger.

The Chain Reaction

At the game's start, all platforms are empty, so four of eight entry paths are blocked. Remember that your turn ends when a ball hits an empty platform and stops. As the switches fill up, the chances increase that a ball will descend through several levels. The goal is to score points by getting balls to pass all the way through the maze of the switchbox. The best way to collect a lot of points is to cause a chain reaction.

A ball that hits a loaded switch from either side continues on its way. And the previously inert ball on the platform starts moving. One enters, two exit. If both of those balls encounter full platforms, four drop from the switches. The pathways are staggered, so the effects can spread outward, with more and more balls cascading toward the bottom.

Rather than taking an easy point or two, it's often worthwhile to build up layers of loaded switches. Watch out for leaving yourself vulnerable, though. Because players take turns, you'll want to leave positions where your opponent's move gives you a chance to create a chain reaction. The best strategy is to play defensively. Look ahead a move or two, and watch for an opening that allows you to score several points at once.

Four Quarters

A game of Switchbox always lasts four rounds. In the first (equality), each exit counts for two points. Your goal is to score ten points. The second quarter has more points available, as well as a higher goal. If you look at the exits, you'll see that the further away from the middle, the higher the point value. The numbers increase in a "Fibonacci" sequence: 1, 2, 3, 5, 8, and so on. Each number is the sum of the previous two (1+2 is 3, 2+3 is 5, 3+5 is 8, etc.). The target score in round two is 40.

Lyco Computer Marketing & Consultants

"WE MAKE YOUR COMPUTER FUN TO USE!"

NO LABEL DISKETTES

NL 5¼" SSDD 10.99 (Box 10)
NL 5¼" DSDD 15.99 (Box 10)
*Free Diskette Writer Pen!
*Free Storage Case!

DUST COVERS

Atari

520ST	11.95
130XE	6.99
800XL	6.99
1050	6.99
1025	7.9[illegible]

Commodore

C128	7.99
1571/1541	6.99
1902	10.95
1702	6.99
C64/Vic20	6.99

Panasonic

1090/1091	6.99
1092	6.99
1093	9.99

Star Micronics

SG/SD10	6.99
SG/SD15	9.99
SR10	9.99
SR15	9.99

Okidata

82/92	6.99
83/93	9.99
193	9.99

PRINTING PAPER

3000 SHEETS FANFOLO	$42.75
1000 SHEETS FANFOLO	$19.75
1000 SHEET LETTER	$21.95
200 SHEETS LETTER	$8.99
150 RAG STATIONARY	$10.99
MAILING LABELS (1in.)	$9.95

WICO Joysticks

15-9714 Bat Handle	16.75
50-2030 Boss	11.99
50-2002 Super 3-Way	19.99

COMMODORE

C-128 NEW	CALL
1571 Drive	CALL
1572 Drive	CALL
1902 Monitor	CALL
1670 Modem	CALL
C-64 Computer	CALL
1541 Drive	189
MPS801 Printer	LOW
1702 Monitor	199
Simon's Basic	24.75
Assembler 64	34.75
Super Expander	22.75
Logo 64	49.75
Pilot 64	38.75

COMMODORE SOFT-WARE

MICROPROSE (C-64)

Kennedy Approach	21.75
Crusade in Europe	24.75
Decision in Desert	24.75
Solo Flight	20.75
Nato Commander	20.75
Spitfire Ace	18.75
F-15 Strike Eagle	20.75
Hellcat Ace	18.75
Acrojet	21.75
Silent Service	21.75

CARDCO

Digitizer Camera	189.95
32K Printer Buffer	59.95
Numeric Keypad	34.95
CB/5 5-slot Board(64)	54.00
CB/2 2-slot Board(64)	25.00
S'More Basic Rom	49.95
Write Now-64	35.00
Mail Now-64	29.00
Spell Now-64	29.00
File Now-64	29.00
Paint Now-64	29.00
Calc Now-64	29.00
Tax Survival	29.00
Super Printer Utility	27.95
Write Now-Vic-20	29.95

BRODERBUND

The Print Shop	28.75
Graphics Library	18.75
Graphics Library II	19.75
Graphics Library III	19.95
Karateka	19.75
Castles Dr. Creep	19.75
Bank St. Writer	32.75
Loderunner	20.75
Mask of the Sun	24.75
Spelunker	19.75
Serpent's Star	24.75
Whistler's Brother	18.75
aid Bungeling Bay	18.75

COMMODORE

MPS1000 Printer	259
C1350 Mouse	42
C1700 128K RAM	145
C1750 512K RAM	269
JANE	35
Perfect Writer	49
Perfect Calc	49
Perfect Filer	49

SPINNAKER (C-64ROM)

Cosmic Life ROM	19.75
Jukebox	19.75
Alphabet Zoo	17.75
Alf in Color Caves	19.75
Up for Grabs	19.75
Delta Drawing	19.75
Kids on Keys	18.75
Kindercomp	14.75
Facemaker	17.75
Fraction Fever	17.75

SSI (C-64)

Colonial Conquest	24.75
Wings of War	24.75
Computer Ambush	34.75
Field of Fire	24.75
Fighter Command	36.75
Kampfgruppe	36.75
Mech Brigade	36.75
Market Garden	29.75
Six Gun Shootout	24.75
Computer Baseball	24.75
Computer Quarterback	24.75
Imperium Galactum	24.75
Phantasia	24.75
Cartels & Cutthroats	24.75
50 Mission Crush	24.75
Questron	24.75

INNOVATIVE CONCEPTS

Flip-n-File 10	3.50
Flip-n-File 15	8.25
Flip-n-File 25 Lock	17.95
Flip-n-File 50	17.25
Flip-n-File 50 Lock	22.95
Flip-n-File Rom	17.25

SCARBOROUGH (C-64)

Build A Book	24.75
Improved Mastertype	23.75
NET WORTH	48.75
Mastertype Filer	22.75
Boston 64 Diet	27.75

TRONIX

S.A.M. - Atari	38.50
S.A.M. - C-64	38.50

PERSONAL PERIPHERALS

Super Sketch 64	32.75
Printer Utility	18.75

BATTERIES INCLUDED

Paper Clip	59.95
Spell Pak	34.95
Consultant	59.95
Paper Clip w/Spell Pak	75.95
Home Pak	34.95
Bus Card	129.95
80 Column Board	109.95

EPYX (C-64)

Fast Load	26.75
Breakdance	23.75
Greatest Baseball	24.75
Summer Games	26.75

SUB LOGIC (C-64)

Flight Simulator II....32.75
Night Mission Pinball 20.75

CONTINENTAL (C-64)

Home Accountant	44.75
1984 Tax Advantage	35.75
1985 C-64 Book of Software	16.95

QR & D

Copy Q.....27.95
GPC Printer Interface 65.00

EASTERN HOUSE

Rabbit C-64	19.95
Rabbit VIC-20	19.95
MAE C-64	27.95
Telstar 64	19.95
M.L. Monitor 64	18.95

KOALA (C-64)

Koala Pad 59.95

COMPUTER CARE

NORTONICS DISK DRIVE CLEANER WITH SOFTWARE

REG. 49.95
NOW 19.95

BUY LYCO AND ENJOY

★ THE LOWEST PRICES ★ TOLL FREE ORDER LINE ★
★ Free shipping on prepaid cash orders in U.S. ★ All Merchandise Factory Fresh ★
★ 24 hrs. shipping on in-stock product ★ Access to our Multi Million $ inventory ★
★ No deposit on UPS C.O.D. orders ★ Orders outside PA save state sales tax ★
★ Air freight service available ★ Full Manufacturer's Warrranty apply! ★ Full accessory line in stock ★
★ Purchase Orders Accepted from educational institutions! ★ We check for stolen credit cards! ★
★ We ship to our servicemen overseas! ★ You'll love our Courteous Sales Staff! ★

AMERICA'S MAIL ORDER HEADQUARTERS

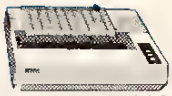
LYCO COMPUTER

WORLD'S LEADER IN SALES & SERVICE

TO ORDER
CALL TOLL FREE
800-233-8760
In PA 1 717-327-1824
Lyco Computer
P.O. Box 5088
Jersey Shore, PA 17740

Lyco Computer Marketing & Consultants

SAVE ON THESE IN STOCK PRINTERS



1091.....$233

AXIOM SEIKOSHA

GP550AT (Atari)	222
GP550CD (C 64	222
GP700AT (Atari)	439
GP700AP (Apple)	439
FI ITE5CD (C 64)	22?

C. ITOH

ProWriter 8510Sp+	349
1550Sp+	489
StarWriter	769
PrintMaster	929

TOSHIBA

P1340	469
P351+	1149
P341P	969
P341S	999
351 Sheet Feeder	529

CARDCO

32K BUFFER (C 64)	59

CORONA

LP300 Laser Printer	2686
200361 Toner Cartridge	89

EPSON

FX85 (New)	333
LX80	212
FX185 (New)	464
LX90 (New)	226
SQ2000 (New)	1555
JX80	467
Homewriter 10	193
CR-20-Atari	153
CR-220-C-64	153
DX-10 (New)	207
DX 20 (New)	297
HS-80 (New)	288
LQ1500P	975
LQ1500S	1039
RX-100	356
FX-100+	CALL

CITIZEN

MSP 10	269
MSP 15	358
MSP 20	337
MSP 25	495
Sheetfdr (10/20)	189
Sheetfdr (15/25)	199

OKIDATA

Okimate 10	179
Okimate 20	CALL
182	214
192	348
193	563
92P	349
93P	565
84P	645
92 Imagewriter	349

*IBM versions also

BROTHER

HR-15XL P	359
HR-15XL S	359
HR 35P	839
HR 35S	839
2024L-P	949
M1009 P	189

JUKI

Juki 6100	347
RS232 Serial Board	55
6100 Tractor	119
6100 Sheet Feeder	209
Juki 6300	757

LEGEND

880	188
1080	222
1380	262
1385	296
LEGEND 808	159

DIGITAL DEVICES

16K BUFFER	75
32K BUFFER	89
64K BUFFER	125

DIABLO

D25	549
630 API	1599
630 ECS	1759
D 80 1F	2395
P 32 CO1	699
P 38	1749
C 150	999
DX 35 (NEW)	CALL
AP 80	CALL

SG-10.....$208

PANASONIC

1091.....	*...233*
3131 (NEW)	269
1092	373
1093	428
3151 Letter	426
4K Buffer	65

SILVER REED

EXP400	249
EXP500	295
EXP550	399
EXP770	749

STAR MICRONICS

SG 10	208
SG 15	373
SD 10	336
SD 15	442
SR 10	483
SR 15	583
SB 10	595
Power Type	303
SG 10 C 64 (NEW)	CALL

MONITORS

TAXAN

115 12" Green Composite	CALL
116 12" Amber Composite	CALL
121 12" Green TTL	135
122 12" Amber TTL	145
220 14" Color Composite	259
410 12" RGB Hi Res IBM	329
420 12" RGB Super Hi IBM	409
440 12" RGB Ultra Hi Res	555
Tilt Stand	35

ZENITH

ZVM 122A Amber	75
ZVM 123G Green	75
ZVM 124 Amber IBM	129
ZVM 131 Color	275
ZVM 133 RGB	369
ZVM 135 Composite	449
ZVM 136 Hi Res Color	589
ZVM 1220	95
ZVM 1230	95
ZVM 1240	149

TEKNIKA

MJ 10 Composite	179
MJ-22 RGB	255

AMDEK

300 Green	118
300 Amber	128
310 Amber IBM	155
Color 300 Audio	234
Color 500 Composite	369
Color 600	397
Color 700	495
Color 710	569

PANASONIC

DT1300D 13" RGB/Composite	247
DTM140 14" RGB/Composite	329
DTH103 10" RGB Hi Res	395
DTS101 10" Composite	175
DT1000G 10" RGB	166
TX12H3P 12" Color	419
TR120M1PA 12" Green	109
TR120MBPA 12" Amber	109
TR122MSP 12" Green IBM	149
TR122MYP 12" Amber IBM	149

SAKATA

SG 1000 12" Green	99
SA 1000 12" Amber	109
SG 1500 12" Green TTL	119
SA 1500 12" Amber TTL	129
SC 100 13" Color Comp	209
SC 200 13" RGB	389
STS1 Tilt Stand	29

X-TRON

Comcolor I Composite Green	177

NEC

JB 1260 Green	95
JB 1201 Green	135
JC 1215 Color	235
JC 1216 RGB	375
JC 1460 Color	265

PRINCETON GRAPHICS

MAX 12 Amber	185
HX 12 RGB	465
SR 12 RGB	595

INTERFACING

DIGITAL DEVICES

U-Print C (C-64)	49

ORANGE MICRO

Grappler CD (C 64)	79

QRD

GPC (C 64)	59

CARDCO

G-Wiz (C-64)	54
C/?+G (C-64)	45
C/?PS (C 64)	49
C/?B (C-64)	39

TYMAC

Connection (C-64)	55

MODEMS

HAYES

Smartmodem 300	133
Smartmodem 1200	377
Smartmodem 1200B	347
Smartmodem 2400	598
Micromodem IIE	135

MICROBITS

1064Modem	57

TELE LEARNING

Total Telecommunications (C-64)	29 95
AP-250 (300 Baud Apple)	69 95
IB-250 (300 Baud IBM)	69 95

US ROBOTICS

Password 1200M	229
Password 1200F	229
Password 300M	139
Password 300F	139
Autodial 212A	359
PCM5	319
PCM64	519
PCM256	769
S 100	379
Courier	469
Microlink	469
Telpac Ms-00S	79

NOVATION

IBM 300/1200 MS-DOS ext	319
IBM 300/1200 CPM-86 ext	319
IBM 300/1200/2400 ext	529
IBM 300/1200/2400 MS-DOS	579
IBM MS-DOS int	325
IBM CPM-86 int	325
Macmodem 300/1200	315
Upgrade Apple Cat II	225
Cat 300 Acoustic	139
J-Cat RS232	89

ANCHOR

Volksmodem	55
Volksmodem 12	186

DRIVES

COMTEL

Enhancer 2000 (c-64) 179

INDUS

GT ATARI	*215*
GT COMMODORE	235

MSD

SD1 Drive (C-64)	229
SD2 Drive (C-64)	469

DISKETTES

DENNISON

ELEPHANT 5¼" SSSD	11.99
ELEPHANT 5¼" SSDD	12.99
ELEPHANT 5¼" DSDD	14.99
PREMIUM 5¼" SSDD	13.99
PREMIUM 5¼" DSDD	15.99

SUNKYONG

SKC 5¼" SSDD	11.99
SKC 5¼" DSDD	13.99

MAXELL

5¼" MD1	13.99

VERBATIM

5¼" SSDD	13.99
5¼" DSDD	19.99

BONUS

5¼" SSDD	9.99
5¼" DSDD	12.99

TOLL FREE 1-800-233-8760

TO ORDER

CALL TOLL FREE

800-233-8760

Customer Service 1-717-327-1825

or send order to
Lyco Computer
P O. Box 5088
Jersey Shore, PA
17740

RISK FREE POLICY

In stock items shipped within 24 hours of order. No deposit on C O D. orders. Free shipping on prepaid cash orders within the continental U S. Volume discounts available. PA residents add sales tax. APO, FPO and international orders add $5 00 plus 3% for priority mail service. Advertised prices show 4% discount for cash; add 4% for MasterCard or Visa. Personal checks require 4 weeks clearance before shipping. Ask about UPS Blue and Red label shipping. All merchandise carried under manufacturer's warranty. Free catalog with order. All items subject to change without notice.

COMPUTE! Books Announces

OUR FIRST EVER

INVENTORY REDUCTION *Sale*

As computers come and go in industry popularity, we try our best to maintain a flow of excellent books for readers and users of the most popular personal computers. You'd be the first to agree that, simply because a particular computer is no longer produced, information on it is no less important to you. But we've found that when some computers lose mass appeal, or are no longer at the top of the current best-seller list, many book stores no longer wish to stock books on them.

These books become arguably more valuable to those who need them . . . those who never got around to buying them . . . or those who have bought their personal computer second-hand, but now can't find books about it.

This sale is for you. It mixes the best of our backlist—from machine-specific to topical titles that never quite caught on—and gives you significant savings on dozens of COMPUTE! titles. Some quantities are very limited, so send in your order soon. Credit card or check with order only. Or call our toll free number: 1-800-346-6767 (in NY 212-887-8525).

Order any three from Group A for $24.95 (an initial savings of at least 30 percent), and receive up to three from Group B for $3.00 each. (A potential *total savings of over $55.00!*) All orders add $2.00 shipping and handling per book up to 5 books. Over 5 books, add $5.00 per order.

Group A (Three for $24.95)

First Book of Atari Graphics
Second Book of Atari Graphics

Commodore 64 Games for Kids
All About the Commodore 64, Vol. 1
First Book of Commodore 64 Sound and Graphics
Reference Guide to Commodore 64 Graphics

Home Computer Wars
Personal Telecomputing
BASIC Programs for Small Computers
Computing Together

Programmer's Reference Guide to the TI-99/4A
TI Games for Kids
33 Programs for the TI-99/4A
Guide to TI-99/4A Sound and Graphics

First Book of VIC
Second Book of VIC
Third Book of VIC
VIC Games for Kids
Programming the VIC

Arcade Games on the Timex/Sinclair
Programmer's Reference Guide to the Color Computer

Group B (Up to three for $3.00 each)

First Book of Atari

First Book of Commodore 64
First Book of Commodore 64 Games
Commodore Peripherals: A User's Guide

First Book of Robots
Home Energy Applications
Beginners Guide to Buying a Personal Computer

First Book of TI Games
Extended BASIC Home Applications on the TI-99/4A
Arcade Games on the TI-99/4A

First Book of VIC Games
Arcade Games on the VIC
Second Book of VIC Games

All sales final. No returns. All are new books in good condition.

Special offer through March 15. Order four books for $34.95 from Group A** and choose up to *six* additional titles from Group B for only $3.00 each.

**substantial savings . . . less than $8.75 each for values up to $24.95.

SAVE 75% or more on selected titles

$1.00 shipping/handling per book for 1–5 books. Over 5 books, $5.00 per order.

Writing An Amiga Game

Philip I. Nelson, Assistant Editor

Writing "Switchbox," our first game translation for the Amiga, posed a number of interesting programming challenges and proved to be an excellent way to become familiar with Microsoft's Amiga BASIC. To show off just a few of the machine's special features, the Amiga version of Switchbox includes fast graphics, stereo sound effects, and voice synthesis.

The first thing you'll notice about the Amiga program is that it has no line numbers. Instead, meaningful labels like *Setup:*, *Putball:* and *Nextround:* mark subroutines and major program divisions. To improve the program's readability, meaningful variable names like *Points*, *Round*, *Column* and *Row* have also been used in a number of cases. If you're familiar with Commodore 128 or 64 BASIC, you may find it interesting to compare one of those versions with this one. Though some routines have been repositioned, and the graphics techniques are very different, this program follows essentially the same logic as the original.

Window Management

Before creating any graphics, you must make some basic decisions about the screen itself. The four PALETTE statements in the *Setup:* routine specify colors for the new screen. If these are omitted, the Amiga uses the same colors that appear when you activate BASIC. The following statement creates a window for the game screen:

```
WINDOW 2,"Switchbox",,0
```

The first parameter (2), creates a new output window specifically for this program's output. If you don't create a new window, all output goes to window 1, which is normally titled with the name of the current program. You could follow this statement with WINDOW OUTPUT 2, to direct all output to window 2. But that's done automatically when you open the window. When you close the window with WINDOW CLOSE 2 (see the *Gohome:* routine), output reverts to window 1 again.

The second parameter in a WINDOW statement is a string that contains the window's title.

The third WINDOW parameter, which is optional, specifies the window's size. Windows can be smaller than the actual screen. In this case, we needed a full-screen window, so we simply left out this parameter. The window automatically expands to the full size of the screen.

The fourth WINDOW parameter also is optional. It specifies the window's type—that is, the window's characteristics. Though it's often desirable to resize a window and move it around the screen, those features aren't needed for a game. To disable them, we specify a window type of 0. This creates a window that can't be resized or moved around with the Title Bar; can't be moved from the front to back of other windows with a Back Gadget; and can't be closed with a Close Gadget. However, Amiga BASIC's normal menus are all left active, so the player can still stop the program by choosing the Stop option from the Run menu.

In any program that includes speech, it's a good idea to include a short SAY statement at the very beginning of the program before opening any custom windows. When the Amiga encounters the first SAY statement, it tries to load the narrator device program from disk (the Amiga's speech synthesizer is implemented in software, not hardware). If it can't find the narrator on the currently mounted disk, it displays a requester box prompting you to insert the correct volume. If this happens *after* you've opened a new window, the requester box may appear on the original output window, which is now invisible. This can be very confusing to a new user, who may think that the system has crashed, when in fact it's just waiting for a response.

Hi-Res Graphics

The 128 version of Switchbox draws the playfield and animates the moving figures with traditional Commodore methods—PRINTing graphic characters on the text screen or POKEing them directly into screen memory. Since the 128's text screen is divided into 25 rows of 40 characters, it's a fairly simple matter to keep everything neatly aligned. Not so on the Amiga, which in this case presents a high-resolution graphics screen 640 pixels wide and 200 pixels high. While it's possible to put characters on this screen with

In round three the numbers are a bit lower. They increase arithmetically (1, 2, 3, 4, up to 8 in the corners). A goal of 20 points brings you to round four, where you can score big. Here the numbers are squares: 1, 4, 9, 16, 25, all the way to 64 at the edges. In rounds two through four, it's sometimes prudent to leave a middle path open for your opponent to score a few points, in order to gather a high score on the big numbers to the left and right.

Each round lasts until one player has reached the goal. At that point the other player has one last turn before the round ends. It's possible to win the round on this last-chance play; watch out for barely topping the goal and leaving a chain reaction open for the other player. An arrow points to the scoreboard of the player whose turn it is. On the other side of the screen, you'll see a number where the arrow should be. That's the goal for the current round (the Amiga version displays the goal on both sides of the screen, below the scoreboards).

Bonus points are awarded at the conclusion of each round. Four

Atari Explodes

Atari's new computer serious threat to Macintosh. Will the Amiga survive?

The Atari 520 ST is a serious challenge to the Apple Macintosh and will open up a major fight in the personal computer market.

By Joseph Sugarman

Imagine this. If I could offer you a Macintosh computer—(a computer that sells for over $2000)—for one third the price, you might wonder.

But what if I offered you a better computer with none of the disadvantages of the Mac and what if I added new features which improved its speed and performance? That's exactly what Atari has done in an effort to grab the ball from Apple and really explode into the personal computer market.

HEADING EFFORT

Heading the effort at Atari is Jack Tramiel—the same man who built Commodore into a billion dollar corporation, sold more computers than any other man in the world and believes in giving the consumer incredible value without sacrificing quality. The new Atari is a perfect example.

First, let's compare the new Atari ST to the Macintosh and the Commodore Amiga. Sorry IBM, we can't compare the ST to your PC because yours is almost five years old, much slower, and, in my judgement, over priced.

Price The cheapest you can get the Macintosh with 512K of memory is $1800 with a one-button mouse, a disk drive and a monochrome monitor. The Amiga sells for $1995 with a two-button mouse, a disk drive and a color monitor. The Atari ST sells for $699 with a two-button mouse, a disk drive and a monochrome monitor and for $200 more, a color monitor. Read on.

Monitor With the Mac you can only use its 9" monochrome monitor and with the Amiga you can only use its 12" color monitor. With the ST you have a choice of either a 12" monochrome or high-resolution color monitor or your own TV set.

Resolution The number of pixels or tiny dots on a screen determine the sharpness of a computer monitor. The Mac has 175,104 pixels and has one of the sharpest screens in the industry. The Atari ST has 256,000 pixels or almost a third more than the Mac. And the Atari color monitor compared to the Amiga in its non interlace mode is 128,000 pixels or exactly the same.

Power All the computers have a 512K memory with a 68000 CPU operating with a 32-bit internal architecture. But Atari uses four advanced custom chips which cause the CPU to run faster and more efficiently giving it some tremendous advantages. For example, it has a faster clock speed of 8Mhz compared to the Mac's 7.83 and the Amiga's 7.16. And the speed of the unit is hardly affected by the memory requirements of the monitor which in the Amiga can eat up much as 70% of the unit's cycle time or speed.

Keyboard This is the part I love. The Mac has a small 59-key keyboard and a mouse. That's all. The 95-key Atari has both a mouse, cursor keys, a numeric keypad and ten function keys. The keyboard looks fantastic and is easy to type on. Although the 89-key Amiga has almost all the features of the Atari keyboard, it looks like a toy in comparison. (Sorry Commodore, but that's my opinion.)

Disk Drive The Mac's 3½" disk drives run at variable speeds—slowing down as they run. The Atari 3½" drives run faster at a constant speed—and quieter than any other unit.

Features The Atari ST comes equipped with the same printer and modem ports as the IBM PC—a parallel and RS232C serial port. The Mac comes only with a tiny non-standard serial and modem port. The ST has a hard disk interface capable of receiving 10 million bits per second. There are two joy stick ports and a 128K cartridge port for smaller programs or games. It has 512 colors (for the color monitor), it has a unique MIDI interface into which you can plug your music synthesizer and record or play back your music.

Software Right now, the Mac has more than the Atari ST and the Amiga combined. The Atari is a new system but the track record of Atari's Jack Tramiel and the potential of the new unit is causing a flood of new software titles. In fact, I'll predict that eventually the Atari will have more software than the Mac. There are now hundreds of titles, from word processing to spread sheet programs, from graphics and games to data base management—all with those easy drop-down menus and windows. There's plenty from which to select now and plenty more to come.

If you think I'm enthusiastic over the ST, listen to what the press is saying. *Byte Magazine* just called it the "Computer of the year for 1986." *Creative Computing* exclaimed, "Without question, the most advanced, most powerful micro computer your money can buy." and finally, the Atari ST is the best selling computer in Europe and acclaimed, "The computer of the year," by the European personal computer press.

I am going to make the ST so easy to test in your home or office that it would be a shame if you did not take advantage of my offer. First, I will offer the computer itself for only $299. You will need, in addition, either one or two disk drives and either an Atari monochrome or color monitor or your own TV. If you order with your credit card during our introduction I will ship your order and only bill you for the postage and 1/3 the purchase price. I will also add a few software packages free including "Logo"—a beginners programming language, a disk for programming in BASIC and Neochrome—a graphics paint program.

COMPARE THE TWO

After you receive the Atari ST, put it next to your Mac or Amiga or even IBM. See how extremely sharp the graphics appear, discover what a perfect word processor it is, how great the keyboard feels and finally how much faster and quieter it runs.

If you're not convinced that the Atari is far superior to your present computer and a fantastic value, simply return it and I'll refund your modest down payment plus our postage and handling charges. If you decide to keep it, I'll bill your credit card account for the remaining balance and enroll you in our discount software club (a $50 value) that lets you buy software for up to 50% off the retail price.

But act fast. We have only 2,000 units and 1,000 free memberships that we will offer as part of this introductory program and we are certain they will go fast. Order today.

To order, credit card holders call toll free and ask for product by number (shown in parentheses). Please add $20 per order for postage and handling. (If you pay by check, you must pay the full amount but we will provide you with a bonus software package.)

ST Keyboard, CPU & Mouse(4060M)	**$299**
Disk Drive (4056M)	**199**
Monochrome Monitor (4057M)	**199**
RGB Color Monitor (4058M)	**399**

Note: A list of software will come with the unit.

JS&A® PRODUCTS THAT THINK

One JS&A Plaza
Northbrook, Illinois 60062

CALL TOLL FREE 800 228-5000

IL residents add 7% sales tax.

LOCATE and PRINT, the Amiga's character set includes no graphics characters. So 128-style graphic techniques are useless unless you want a game screen that consists of X's, O's, and slash characters. Instead, everything must be drawn with hi-res commands like LINE, PUT, CIRCLE, and PAINT (see the routine labeled *Setup:*).

Repeated shapes are stored in an array with GET and then placed in several locations with PUT commands. PUT is used extensively in this game to create the moving balls, switches, and arrows, as well as to draw parts of the switchbox itself. Though it's one of Amiga BASIC's slower graphics commands, PUT is more than adequate for a game of this type and much faster than the same commands in other BASICs. Thanks to the Amiga's fast 68000 microprocessor and custom graphics chips, this version runs much faster than the original, even though no particular attempt was made to optimize the program's speed.

PUT has several different modes which determine what happens when you PUT a shape into an area that already contains graphics data. The Amiga uses XOR (exclusive or) mode for PUT unless you specify otherwise. This mode is particularly useful for animation, since if you PUT the same shape twice into the same location with XOR, it erases itself without disturbing whatever was there before. Here's a typical use of PUT with XOR:

```
PUT (140,5),Larrow:PUT (440,5),Rarrow
```

These statements, found in the *Taketurn:* routine, are performed on each new turn to make the player's arrows flip back and forth. If you're not familiar with XOR mode, this code looks confusing, since it PUTs both arrows on the screen no matter whose turn it is. But we began the game by PUTting the left arrow in place when the screen was drawn. Thus, when the first turn is taken, the left arrow erases itself, and the right one appears. On the second turn the right arrow erases itself, and the left one appears, and so on. This shortcut eliminates the need for a separate routine to keep track of the arrows' display status.

Speech And Stereo Sound

Speech and stereo sound effects may seem flashy, but the Amiga makes them quite easy to program. Amiga BASIC's SAY TRANSLATE$ command translates any English text into quite understandable speech. And it's easy to flip sound effects from one stereo output to another by changing the final value in a SOUND statement.

If your monitor has only one speaker, you'll probably want to defeat the stereo feature so that both players' sound effects can be heard through one output. This is easy to do. The Amiga has four sound channels, numbered 0–3: Channels 0 and 3 always go to the left speaker, and channels 1 and 2 go to the right. There are five SOUND statements in the program (in the routines labeled *Switch:*, *Putball:* and *Score:*). In each SOUND statement, the final parameter controls whether the sound goes to the left or right output. For instance, the *Score:* routine contains these statements:

```
SOUND j,1,64,Who
SOUND j+400,1,64,3-Who
```

The program variable Who equals 0 when it's the left player's turn, and 1 on the right player's turn. Thus, when Who=0, the expressions Who and 3−Who create sound in channels 0 and 3, which both go to the left speaker. When Who=1, they output through channels 1 and 2, which go to the right speaker. To defeat the stero effect in these statements, replace Who and 3−Who with the values 0 and 3, or 1 and 2, depending on which output you're using. Similar changes to the other SOUND statements will confine them to one speaker as well.

If you don't specify otherwise, SAY commands cause the program to halt until the computer finishes saying the current phrase. But at certain points in Switchbox, the computer talks "in the background" while it performs other program tasks. At the beginning, for example, you'll see it draw several graphics shapes while it pronounces the welcoming phrase. This effect is also quite simple to achieve.

Look at the first set of DATA statements in the *Setup:* routine: These values are stored in an integer array (Voice%) for later use in SAY commands. Each element of the voice array controls a different aspect of the Amiga's speech, such as pitch, speaking rate, and so on. The next-to-last element in the array controls whether the program continues while SAY commands are in progress. Setting this value to 1 selects *synchronous* speech which proceeds in the background. Replacing the 1 with a 0 selects *asynchronous* speech mode, which halts program execution until the current phrase is finished.

numbers appear below the scorecards. The first is simply the total so far. The second is the total plus a bonus of the goal for the round if the player's points are equal to or greater than the goal. For example if the goal is 20 and you get 18, there's no bonus. If you score 22, the bonus is the goal for that round (20) and you'd have 42 points. The third number under the scoreboard is the difference between scores for the rounds. If you win by two points, two is added to your score (and two is subtracted from the other player). The final number is the grand total of the first three scores and bonuses. Rounds one and three are fairly low-scoring with low goals. You may want to seed the field with extra balls during these quarters, so you can collect more points in the second and fourth quarters.

Variations

Although the goal of the game is to score the most points, there's no reason you couldn't agree to play for low score. In a "lowball" game, you would try to avoid scoring points. You wouldn't necessarily play backwards, you would have to adjust the strategy of where to place the balls. Fill up the board as much as possible and leave your opponent in a situation where he or she

is forced to score points.

The DATA statements at the beginning of the program (the *Set-up*: routine in the Amiga version) determine the goal for each round and the point values for the exit paths. You can prolong the game by doubling the goals; this also dilutes the value of a big score at the beginning of a round, preventing one player from winning on the first or second turn. An interesting variation is to assign negative values to some slots. If some paths score negative points, you are forced to think harder about where the balls will drop.

In addition to the numbered keys (1–8), the plus (+) and minus (−) keys are active. Pressing plus drops a ball at random down one of the eight entry paths. Pressing minus allows you to pass your turn to your opponent.

Once you've mastered the regular game, you can add some new rules. Each player gets three passes per half, similar to the three timeouts in a football game. If you don't like the looks of the board, press the minus key to use one of your passes. After one player has skipped a turn, the other player must play (this prevents the possibility of six passes in a row). It's also a good idea to make a rule that a player can't pass on two consecutive turns. You can also give each player two random moves to be played for the opponent. In other words, after making a move, you could inform your opponent that you're going to give him one of your random moves and you would press the plus key.

Here's one more change you could make: Instead of alternating turns, allow a player to continue after scoring. When a player drops a ball and scores some points, the other player would have to pass (by pressing the minus key). If the first player scores again, the opponent passes again, and so on until no more points are scored.

Playing Solitaire

To drop a ball, press a numbered key (1–8). If you're using a 128, ST or Amiga, the numeric keypad is convenient for choosing a move. By using the pass and random turn options, you can play against the computer. Here are the rules for solitaire play:

1. The computer always scores first. At the beginning of every round, the computer plays randomly until at least one point is acquired. Press the plus key for the computer's turn. You must continue passing (skip your turn with the minus key) until the computer puts points on the board.

2. After the first score by the computer, you can begin to play. When the computer has a turn, press the plus key for a random move.

3. Whenever you make points, you must pass again until the computer scores. When the computer gets more points, you can begin to play again. This rule means you should hold back on the easy scores of a few points; wait until there's an avalanche available.

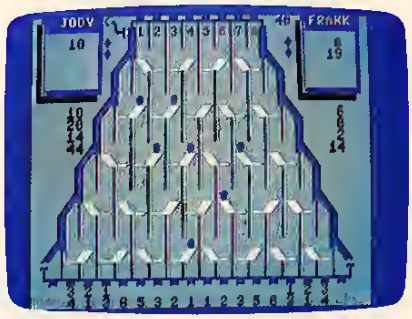
4. If you're the first to reach the goal, the computer gets a last chance. Don't make this move randomly; figure out the best opportunity for scoring and play that move for the last-chance turn.

In the interest of keeping these programs to a manageable length, no attempt has been made to provide an "intelligent" computer opponent. Once you become familiar with the game, you might find it an interesting project to try adding some routines that give the computer a rational basis for picking one move over another.

For instructions on entering these listings, please refer to "The New Automatic Proofreader for Commodore" and "COMPUTE!'s Guide to Typing In Programs" in this issue of COMPUTE!.

Program 1. Commodore 128 Switchbox

```
FP 10 DIMSW(4,7,1),SP$(1),LB(3
      2,4),AR$(1),PT(4,16),SC(
      1,8)
DE 12 SP$(0)="{OFF}[*]{RVS}[*]
      {OFF}[@]":SP$(1)="[@]
      {RVS}£{OFF}£":AR$(0)="
      <I {DOWN}{2 LEFT}J[W]":A
      R$(1)="[Q]K{UP}{2 LEFT}
      {SPACE}U>":QR=1:PRINTCHR
      $(27);"M"
EC 14 COLOR0,16:COLOR4,7:COLOR
      5,7:TX=RND(-TI/137)
QS 20 FORJ=1TO4:READPT(J,0):RE
      M NAME AND GOAL
XC 22 FORK=1TO8:READL:PT(J,K+8
      )=L:PT(J,9-K)=L:NEXTK,J:
      REM POINTS
RP 24 DATA 10:REM ROUND 1 (EQU
      AL)
PE 25 DATA 2,2,2,2,2,2,2,2
PF 26 DATA 40:REM ROUND 2 (FIB
      ONACCI)
HP 27 DATA 1,2,3,5,8,13,21,34
KJ 28 DATA 20:REM ROUND 3 (ARI
      THMETIC)
8G 29 DATA 2,3,4,5,6,7,8,9
E8 30 DATA 80:REM ROUND 4 (SQU
      ARES)
SF 31 DATA 1,4,9,16,25,36,49,6
      4
P8 40 SCNCLR:INPUT"PLAYER 1";P
      1$:INPUT"PLAYER 2";P2$:P
      1$=LEFT$(P1$,5):P2$=LEFT
      $(P2$,5):PRINTP1$;" VS "
      ;P2$
JD 42 PRINT"IS THIS CORRECT?":
      GETKEYA$:IFASC(A$)<>89TH
      EN40
P8 50 GOSU8500:GOSU8700:REM SE
      TUP
HD 60 FORRR=1TO4:TX=1072+40*RR
      :POKETX,90:POKETX+22,90
XG 62 GOSU8620:REM PUT SCORES
      {SPACE}AT BOTTOM
SF 65 QR=1-QR:COLOR5,7:TY=QR*2
      0:TX=28-TY:WINDOWTX,0,TX
      +2,1,1:PRINTRIGHT$(STR$(
      PT(RR,0)),3):PRINT"
      {2 HOME}":TX=8+TY:CHAR1,
      TX,QR,AR$(QR)
SK 70 GOSUB900:IFSC(1-QR,RR)=>
      PT(RR,0)THEN300:REM END
      {SPACE}OF ROUND
AK 80 GOTO65
EX 300 FORJ=0TO1:FORK=5TO8:SC(
      J,K)=0:NEXTK,J
OP 310 FORJ=0TO1:FORK=1TO4:GL=
      PT(K,0):AC=SC(J,K):SC(J
      ,5)=SC(J,5)+AC:SC(J,6)=
      SC(J,6)-(AC=>GL)*GL:SC(
      J,7)=SC(J,7)+(SC(J,K)-S
      C(1-J,K)):NEXTK,J
OB 320 FORJ=0TO1:FORK=6TO7:SC(
      J,K)=SC(J,K)+SC(J,5):NE
      XTK,J
SC 330 FORJ=0TO1:FORK=5TO7:SC(
      J,8)=SC(J,8)+SC(J,K):NE
      XTK,J
ME 340 COLOR5,12:FORJ=0TO1:FOR
      K=5TO8:Y$=STR$(SC(J,K))
      :L=LEN(Y$):TX=6+J*31-L:
      TY=3+K:CHAR1,TX,TY,Y$:N
      EXTK,J
CX 400 NEXTRR:REM END OF MAIN
      {SPACE}LOOP 60-499
EB 499 GETKEYA$:RUN
8A 500 SCNCLR:PRINTSPC(11);"
      [A]{RVS}[O]{OFF}";:FORJ
      =1TO7:PRINT"{OFF}[R]
      {RVS}[O]";:NEXT:PRINT"
      {OFF}[S]":LL=7
QB 510 FORJ=0TO4:TX=9-2*J:TY=1
```



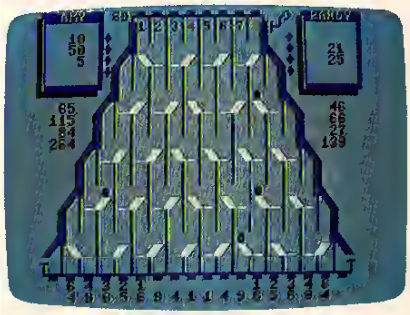
"Switchbox" for the Commodore 128, a challenging strategy game.

```
      +J*4:BX=TX+20+J*4:BY=TY
      +4:WINDOWTX,TY,BX,BY:R$
      ="- "
CD 520 FORK=1TO2:PRINT"
      {2 SPACES}{RVS} {OFF} "
      ;:GOSUB600:PRINT"{RVS}
      {SPACE}":NEXT
8Q 530 PRINT" {RVS}£ {OFF} ";
      :GOSUB600:PRINT"{RVS}
      £*£"
PM 540 LL=LL+2:PRINT"{RVS}£
      {OFF}£";:GOSUB600:PRIN
      T"{LEFT}£*£{RVS}£*£
      {OFF}";:NEXTJ
JP 550 WINDOW1,21,38,23:PRINT"
      £R£ ";:GOSUB600:PRINT"
      £R£"
BF 560 R$="{RVS}£U£{OFF}£E£":L
      L=LL+1:PRINT"£Z£";:GOSU
      B600:PRINT"{LEFT}£X£":W
      INDOW0,0,39,24
QS 599 RETURN
KX 600 FORL=1TOLL:PRINTR$;:NEX
      T:RETURN
MA 620 COLOR5,12:FORJ=1TO16:K=
      PT(RR,J):JJ=2+J*2
RK 630 IFK>9THENL=1NT(K/10):L$
      =M1D$(STR$(L),2,1):ELSE
      L$=CHR$(32)
MC 640 CHAR1,JJ,23,L$:CHAR1,JJ
      ,24,RIGHT$(STR$(K),1):N
      EXTJ:RETURN
SX 700 FORJ=0TO4:SY=4+J*4:FORK
      =0TOJ+3:SX=12-J*2+K*4:C
      HAR1,SX+1,SY-1," "
MX 710 WP=INT(RND(1)*2)
HA 720 SW(J,K,0)=WP:SW(J,K,1)=
      0:GOSUB800
RM 730 NEXTK,J
SK 740 FORJ=1TO8:POKE1074+J*2,
      48+J:NEXT
XJ 750 FORJ=0TO1:8X=J*31:WINDO
      W8X,0,BX+7,7
8Q 760 PRINT"{OFF}{8LK}£D£
      {RVS}{PUR}{7 SPACES}
      {BLK}£K£{PUR}£D£{5 I}
      £F£";
EQ 770 FORK=1TO4:PRINT"{RVS}
      {8LK}£K£{OFF}{PUR}£K£
      {5 SPACES}{RVS}£K£";:NE
      XT
KQ 775 PRINT"{RVS}{BLK}£K£
      {PUR}£C£{OFF}£5 I£{RVS}
      £V£{OFF}{BLK}£C£{RVS}
      £6 I£{OFF}£V£";
HK 780 NEXT:PRINT"{2 HOME}":CO
      LOR5,5
RE 790 CHAR1,3+(LEN(P1$)=5),0,
      P1$,1
QJ 791 CHAR1,34+(LEN(P2$)=5),0
      ,P2$,1
RP 799 RETURN
BA 800 COLOR5,2:CHAR1,SX,SY,SP
      $(WP):RETURN
JJ 900 FORJ=0TO32:LB(J,0)=0:NE
      XT:NB=1:POKE208,0
RC 910 GETKEYA$:IFA$="-"THENRE
      TURN:ELSEIFA$="+"THENA$
      =STR$(INT(RND(1)*8+1))
FX 915 A=VAL(A$):IF(A<1)OR(A>8
      )THEN910
FK 920 LB(0,0)=1:FORJ=1TO3:LB(
      0,J)=0:NEXT:LB(0,4)=10+
      A*2
SF 1000 DO:EX=1
KR 1010 FORJ=0TO32:IFLB(J,0)TH
      ENEX=0:GOSUB1100
GP 1020 NEXT:IFEXTHENEXIT
EF 1030 LOOP:RETURN
KJ 1100 DY=LB(J,0):DX=LB(J,1):
      LY=LB(J,2):NY=LB(J,3):
      NX=L8(J,4):SM=1064+NX+
      LY*160+NY*40:IF(LY+NY)
      THENPOKESM,32
GJ 1110 LB(J,3)=(NY+1)AND3:ONN
      Y+1GOTO1200,1300,1400,
      1500
EE 1200 IFLY>4THENLB(J,0)=0:GO
      TO1700:REM SCORING ROU
      TINE
QE 1220 POKESM+40,81:ONINT(RND
      (1)*3+1)GOTO1800,1810,
      1820
QS 1300 VX=0:GOSUB1600:IF SW(W
      Y,WX,1)AND(SW(WY,WX,0)
      =SD)THEN VX=1-2*SD:L8(
      J,1)=VX:LB(J,3)=NY+1:L
      B(J,4)=NX+VX:POKESM+40
      +VX,81:GOTO1840
EG 1310 IF SW(WY,WX,0)=SDTHENL
      8(J,0)=0:SW(WY,WX,1)=1
      :POKE SM+40,81:GOTO183
      0
HC 1320 LB(J,3)=NY+1:POKESM+40
      ,81:ONINT(RND(1)*3+1)G
      OTO1800,1810,1820
QD 1400 L8(J,1)=0:L8(J,4)=NX+D
      X:POKESM+40+DX,81:GOTO
      1850
FD 1500 L8(J,2)=LY+1:POKESM+40
      ,81:GOSUB1600:SW(WY,WX
      ,0)=1-SW(WY,WX,0)
DA 1510 IF SW(WY,WX,1)THENL8(N
      8,0)=1:L8(NB,1)=0:L8(N
      B,2)=LY:LB(N8,3)=0:LB(
      NB,4)=NX+2-SD*4:NB=NB+
      1:SW(WY,WX,1)=0:POKESM
      -40+2-SD*4,32:GOSUB186
      0
PA 1520 SX=12-WY*2+WX*4:SY=4+W
      Y*4:WP=SW(WY,WX,0):GOS
      UB800:GOTO1840
FH 1600 WY=LY:JX=(NX/2)+LY-6:W
      X=INT(JX/2):SD=JXAND1:
      RETURN
KX 1700 SF=PT(RR,NX/2-1)
RA 1710 SG=SC(QR,RR)+SF:COLOR5
      ,12
GG 1720 TX=5+31*QR+(SG>9)+(SG>
      99)+(SG>999)
QS 1730 TY=1+RR:A$=MID$(STR$(S
      G),2)
JJ 1740 CHAR1,TX,TY,A$:SC(QR,R
      R)=SG:GOTO1870
MJ 1800 SOUND1,4500,8:RETURN
CP 1810 SOUND1,9000,8:RETURN
FC 1820 SOUND1,6750,8:RETURN
AH 1830 SOUND2,7500,8,1,6250,1
      25,1,1024:RETURN
QD 1840 SOUND2,6000,12,2,4200,
      150,3:RETURN
EH 1850 SOUND2,30000,12,2,1000
      0,5000,3:RETURN
8X 1860 SOUND3,1500,24,0,1450,
      25,3:RETURN
RQ 1870 SOUND1,12000,24:SOUND2
      ,7500,12,0,7300,25:SOU
      ND3,9000,18:RETURN
```

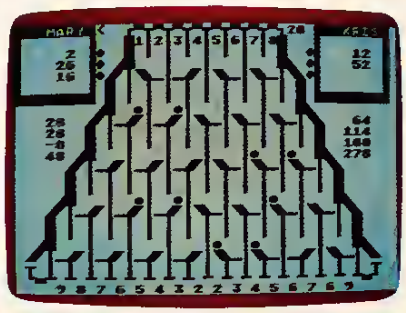
Program 2. Commodore 64 Switchbox

```
R8 100 FORA=54272TO54295:POKEA
      ,0:NEXT:POKE54296,15:PO
      KE54277,24:POKE54284,26
HE 110 V=54276:LB=54272:HB=L8+
      1
GP 120 DIMSW(4,7,1),SP$(1),LB(
      32,4),AR$(1),PT(4,16),S
      C(1,8)
RP 130 SP$(0)="{OFF}£*£{RVS}
      £*£{OFF}£@£":SP$(1)="
      £@£{RVS}£{OFF}£":AR$(
      0)="<I {DOWN}{2 LEFT}J
      £W£":AR$(1)="£Q£K{UP}
      {2 LEFT} U>":QR=1
CF 140 POKE53281,15:POKE53280,
      15:POKE646,6:TX=RND(-TI
      /137)
BD 150 FORJ=1TO4:READPT(J,0):R
      EM NAME AND GOAL
SQ 160 FORK=1TO8:READL:PT(J,K+
      8)=L:PT(J,9-K)=L:NEXTK,
      J:REM POINTS
HH 170 DATA 10:REM ROUND 1 (EQ
      UAL)
EE 180 DATA 2,2,2,2,2,2,2,2
RX 190 DATA 40:REM ROUND 2 (FI
      BONACCI)
PH 200 DATA 1,2,3,5,8,13,21,34
MD 210 DATA 20:REM ROUND 3 (AR
      ITHMETIC)
KP 220 DATA 2,3,4,5,6,7,8,9
SQ 230 DATA 80:REM ROUND 4 (SQ
      UARES)
ED 240 DATA 1,4,9,16,25,36,49,
      64
XS 250 PRINT"{CLR}":INPUT"PLAY
      ER 1";P1$
PD 260 INPUT"PLAYER 2";P2$:P1$
      =LEFT$(P1$,5):P2$=LEFT$
      (P2$,5):PRINTP1$;" VS "
      ;P2$
BH 270 PRINT"IS THIS CORRECT?"
      :POKE198,0:WAIT198,1:GE
      TA$:IFASC(A$)<>89THEN25
      0
EF 280 GOSU8450:GOSUB610:REM S
      ETUP
KP 290 FORRR=1TO4:TX=1072+40*R
      R:POKETX,90:POKETX+22,9
      0
HG 300 GOSUB560:REM PUT SCORES
       AT BOTTOM
RK 310 QR=1-QR:POKE646,6:TY=QR
      *20:TX=28-TY:CX=TX:CY=0
GJ 320 M$=RIGHT$(STR$(PT(RR,0)
      ),3)+"{DOWN}{3 LEFT}
      {3 SPACES}":GOSU81180
GE 330 TX=8+TY:CX=TX:CY=QR:M$=
      AR$(QR):GOSU81180
EA 340 GOSUB770:IFSC(1-QR,RR)=
      >PT(RR,0)THEN360:REM EN
      D OF ROUND
MJ 350 GOTO310
QP 360 FORJ=0TO1:FORK=5TO8:SC(
      J,K)=0:NEXTK,J
HH 370 FORJ=0TO1:FORK=1TO4:GL=
      PT(K,0):AC=SC(J,K):SC(J
      ,5)=SC(J,5)+AC
FF 380 SC(J,6)=SC(J,6)-(AC=>GL
      )*GL:SC(J,7)=SC(J,7)+(S
      C(J,K)-SC(1-J,K)):NEXTK
      ,J
AP 390 FORJ=0TO1:FORK=6TO7:SC(
      J,K)=SC(J,K)+SC(J,5):NE
      XTK,J
XS 400 FORJ=0TO1:FORK=5TO7:SC(
      J,8)=SC(J,8)+SC(J,K):NE
      XTK,J
RJ 410 POKE646,11:FORJ=0TO1:FO
      RK=5TO8:Y$=STR$(SC(J,K)
      ):L=LEN(Y$):TX=6+J*31-L
HE 420 TY=3+K:CX=TX+(TX<20):CY
      =TY:M$=Y$:GOSU81180:NEX
      TK,J
SC 430 NEXTRR:REM END OF MAIN
      {SPACE}LOOP 60-499
XJ 440 POKE198,0:WAIT198,1:RUN
PQ 450 PRINT"{CLR}";:PRINTSPC(
      11);"£A£{RVS}£O£{OFF}";
      :FORJ=1TO7:PRINT"{OFF}
```

```
      {R}{RVS}{O}";:NEXT:PRIN
      T"{OFF}{S}":LL=7
CQ 460 FORJ=0TO4:TX=9-2*J:TY=1
      +J*4:R$="_ "
CJ 470 FORK=1TO2:CX=TX:CY=TY+K
      -1:M$="":GOSUB11B0
QP 480 PRINT"{2 SPACES}{RVS}
      {OFF} ";:GOSUB550:PRINT
      "{RVS} ":NEXT
MB 490 CX=TX:CY=TY+K-1:M$="":G
      OSUB1180
XD 500 PRINT" {RVS}£ {OFF} ";
      :GOSUB550:PRINT"{RVS}
      {*}":CX=TX:CY=TY+K:M$="
      ":GOSUB1180
CA 510 LL=LL+2:PRINT"{RVS}£
      {OFF}£";:GOSUB550:PRIN
      T"{LEFT}{*}{RVS}{*}
      {OFF}";:NEXTJ
MK 520 PRINT:PRINT"{RIGHT}{R}
      {SPACE}";:GOSUB550:PRIN
      T"{R}"
CD 530 R$="{RVS}{U}{OFF}{E}":L
      L=LL+1:PRINT" {Z}";:GOS
      UB550:PRINT"{LEFT}{X}"
QJ 540 RETURN
AJ 550 FORL=1TOLL:PRINTR$;:NEX
      T:RETURN
SX 560 POKE646,11:FORJ=1TO16:K
      =PT(RR,J):JJ=2+J*2
MM 570 IFK>9THENL=INT(K/10):L$
      =MID$(STR$(L),2,1):GOTO
      590
GB 580 L$=CHR$(32)
QH 590 CX=JJ:CY=23:M$=L$:GOSUB
      11B0:CX=JJ:CY=24:M$=RIG
      HT$(STR$(K),1):GOSUB11B
      0
KB 600 NEXTJ:RETURN
XJ 610 FORJ=0TO4:SY=4+J*4:FORK
      =0TOJ+3:SX=12-J*2+K*4
SR 620 CX=SX+1:CY=SY-1:M$=" ":
      GOSUB1180
XG 630 M$=" ":CX=SX+1:CY=SY-1:
      GOSUB1180:WP=INT(RND(1)
      *2)
DM 640 SW(J,K,0)=WP:SW(J,K,1)=
      0:GOSUB760
SH 650 NEXTK,J
SC 660 FORJ=1TOB:POKE1074+J*2,
      48+J:NEXT
JS 670 FORJ=0TO1:BX=J*31:CX=BX
      :CY=0:M$="":GOSUB1180
AG 680 PRINT"{OFF}{BLK}{D}.
      {RVS}{PUR}{7 SPACES}
      {DOWN}{B LEFT}{RVS}
      {BLK}{K}{PUR}{D}{5 I}
      {F}";
DQ 690 FORK=1TO5:CX=BX:CY=K:M$
      ="":GOSUB11B0:PRINT"
      {RVS}{BLK}{K}{OFF}{PUR}
      {K}{5 SPACES}{RVS}{K}":
      NEXT
DA 700 CX=BX:CY=K:M$="":GOSUB1
      180
GS 710 PRINT"{RVS}{BLK}{K}
      {PUR}{C}{OFF}{5 I}{RVS}
      {V}{DOWN}{B LEFT}{OFF}
      {BLK}{C}{RVS}{6 I}{OFF}
      {V}"
XC 720 NEXT:POKE646,4
PD 730 CX=3+(LEN(P1$)=5):CY=0:
      M$="{RVS}"+P1$+"{OFF}":
      GOSUB1180
HP 740 CX=34+(LEN(P2$)=5):CY=0
      :M$="{RVS}"+P2$+"{OFF}"
      :GOSUB1180
KE 750 RETURN
RC 760 POKE646,1:CX=SX:CY=SY:M
      $=SP$(WP):GOSUB1180:RET
      URN
MD 770 FORJ=0TO32:LB(J,0)=0:NE
      XT:NB=1
AG 7B0 POKE198,0:WAIT19B,1:GET
      A$
RA 790 IFA$="-"THENRETURN
MX 800 IFA$="+"THENA$=STR$(INT
      (RND(1)*8+1))
HC 810 A=VAL(A$):IF(A<1)OR(A>B
      )THEN7B0
SH 820 LB(0,0)=1:FORJ=1TO3:LB(
      0,J)=0:NEXT:LB(0,4)=10+
      A*2
EX 830 EX=1
RM B40 FORJ=0TO32:IFLB(J,0)THE
      NEX=0:GOSUB870
BB 850 NEXT:IFEXTHENRETURN
DR 860 GOTOB30
AM B70 DY=LB(J,0):DX=LB(J,1):L
      Y=LB(J,2):NY=LB(J,3):NX
      =LB(J,4)
SF 8B0 SM=1064+NX+LY*160+NY*40
      :IF(LY+NY)THENPOKESM,32
EF 890 LB(J,3)=(NY+1)AND3:ONNY
      +1GOTO900,920,960,970
BC 900 IFLY>4THENLB(J,0)=0:GOT
      01030:REM SCORING ROUTI
      NE
MS 910 POKESM+40,81:ONINT(RND(
      1)*3+1)GOTO1080,1090,11
      00
XR 920 VX=0:GOSUB1020:IF SW(WY
      ,WX,1)=0OR(SW(WY,WX,0)=
      SD)=0THEN940
HP 930 VX=1-2*SD:LB(J,1)=VX:LB
      (J,3)=NY+1:LB(J,4)=NX+V
      X:POKESM+40+VX,81:GOTO1
      120
HB 940 IF SW(WY,WX,0)=SDTHENLB
      (J,0)=0:SW(WY,WX,1)=1:P
      OKE SM+40,81:GOTO1110
DP 950 LB(J,3)=NY+1:POKESM+40,
      81:ONINT(RND(1)*3+1)GOT
      010B0,1090,1100
JS 960 LB(J,1)=0:LB(J,4)=NX+DX
      :POKESM+40+DX,81:GOTO11
      30
MQ 970 LB(J,2)=LY+1:POKESM+40,
      81:GOSUB1020:SW(WY,WX,0
      )=1-SW(WY,WX,0)
AH 9B0 IF SW(WY,WX,1)=0THEN101
      0
MD 990 LB(NB,0)=1:LB(NB,1)=0:L
      B(NB,2)=LY:LB(NB,3)=0:L
      B(NB,4)=NX+2-SD*4:NB=NB
      +1
AC 1000 SW(WY,WX,1)=0:POKESM-4
      0+2-SD*4,32:GOSUB1140
MC 1010 SX=12-WY*2+WX*4:SY=4+W
      Y*4:WP=SW(WY,WX,0):GOS
      UB760:GOTO1120
FA 1020 WY=LY:JX=(NX/2)+LY-6:W
      X=INT(JX/2):SD=JXAND1:
      RETURN
SB 1030 SF=PT(RR,NX/2-1)
GE 1040 SG=SC(QR,RR)+SF:POKE64
      6,11
RJ 1050 TX=5+31*QR+(SG>9)+(SG>
      99)+(SG>999)
EJ 1060 TY=1+RR:A$=MID$(STR$(S
      G),2)
BE 1070 CX=TX:CY=TY:M$=A$:GOSU
      B1180:SC(QR,RR)=SG:GOT
      01150
KK 1080 POKELB,48:POKEHB,4:POK
      E2V,32:POKEV,33:RETURN
QA 1090 POKELB,97:POKEHB,8:POK
      EV,32:POKEV,33:RETURN
BA 1100 POKELB,152:POKEHB,5:PO
      KEV,32:POKEV,33:RETURN
FA 1110 POKEV,32:POKEV,33:FORA
      =50TO10STEP-1:POKEHB,A
      :NEXT:RETURN
HH 1120 RETURN
PB 1130 POKELB,152:POKEHB,10:P
      OKEV,12B:POKEV,129:RET
      URN
RM 1140 POKELB+7,0:POKEHB+7,2:
      POKEV+7,128:POKEV+7,12
      9:RETURN
DR 1150 POKELB,195:POKEHB,16:P
      OKELB+7,135:POKEHB+7,3
      3:POKEV,32:POKEV,33:PO
      KEV+7,32
QX 1160 POKEV+7,33:RETURN
HE 1170 REM CHAR COMMAND
FP 1180 POKE783,0:POKE781,CY:P
      OKE7B2,CX:SYS65520:PRI
      NTM$;:RETURN
```



The Commodore 64 version of "Switchbox" makes good use of character graphics.



"Switchbox" for eight-bit Atari computers.

Program 3. Atari Switchbox

Version by Kevin Mykytyn, Editorial Programmer

```
HI 100 OPEN #1,4,0,"K:":SCR=
       PEEK(88)+256*PEEK(89)
       :POKE 82,0:POKE 752,0
AL 120 DIM SW(4,7),SX(4,7),S
       P$(6),L8(32,4),AR$(6)
       ,PT(4,16),SC(1,8),P1$
       (20),P2$(20)
AB 125 DIM M$(20),T$(20),Y$(
       10),R$(10),L$(10),A$(
       5)
GA 127 FOR A=0 TO 1:FOR B=0
       TO 8:SC(A,B)=0:NEXT B
       :NEXT A
NJ 130 SP$(1,3)="{■}{J}{N}":
       SP$(4,6)="{N}{H}{I}":
       AR$(1,3)="<   ":AR$(4,
       6)="   >":QR=1
BG 140 SETCOLOR 4,3,2:SETCOL
       OR 2,0,8:SETCOLOR 1,0
       ,0
EL 150 FOR J=1 TO 4:READ Q:P
       T(J,0)=Q:REM NAME AND
```

```
        GDAL
PB 160  FDR K=1 TO B:READ L:P
        T(J,K+B)=L:PT(J,9-K)=
        L:NEXT K:NEXT J:REM P
        OINTS
BD 170  OATA 10
HH 180  DATA 2,2,2,2,2,2,2,2
BI 190  OATA 40
BB 200  OATA 1,2,3,5,8,13,21,
        34
AP 210  OATA 20
ID 220  OATA 2,3,4,5,6,7,8,9
BH 230  OATA 80
JA 240  OATA 1,4,9,16,25,36,4
        9,64
JI 250  PRINT "{CLEAR}":PRINT
         "PLAYER 1 ";:INPUT P
        1$
NK 260  PRINT "{DDWN}PLAYER 2
        ";:INPUT P2$:IF LEN(
        P1$)>5 THEN P1$=P1$(1
        ,5)
AP 264  IF LEN(P2$)>5 THEN P2
        $=P2$(1,5)
KA 266  PRINT :PRINT P1$;" VS
        ";P2$
MA 270  PRINT "{DDWN}IS THIS
        CORRECT?":GET #1,A:IF
         CHR$(A)<>"Y" THEN 25
        0
BH 280  POKE 752,1:GOSUB 450:
        GDSUB 610:REM SETUP
FJ 290  FOR RR=1 TO 4:TX=SCR+
        4B+40*RR:POKE TX,96:P
        OKE TX+22,96
PD 300  GOSUB 560:REM PUT SCO
        RES AT BDTTOM
DA 310  QR=1-QR:TY=QR*20:TX=2
        B-TY:CX=TX:CY=0
LF 320  M$=STR$(PT(RR,0)):M$(
        3,3)=" "
GD 330  GOSUB 1180:TX=B+TY:CX
        =TX:CY=0:M$=AR$(QR*3+
        1,QR*3+3):GOSUB 1180
IA 340  GOSUB 770:IF SC(1-QR,
        RR)>=PT(RR,0) THEN 36
        0:REM ENO DF RDUNO
BF 350  GOTD 310
JK 360  FOR J=0 TO 1:FOR K=5
        TD B:SC(J,K)=0:NEXT K
        :NEXT J
FA 370  FDR J=0 TO 1:FOR K=1
        TD 4:GL=PT(K,0):AC=SC
        (J,K):SC(J,5)=SC(J,5)
        +AC
EG 380  SC(J,6)=SC(J,6)+(AC>=
        GL)*GL:SC(J,7)=SC(J,7
        )+(SC(J,K)-SC(1-J,K))
        :NEXT K:NEXT J
NC 390  FOR J=0 TD 1:FOR K=6
        TO 7:SC(J,K)=SC(J,K)+
        SC(J,5):NEXT K:NEXT J
LJ 400  FOR J=0 TO 1:FDR K=5
        TO 7:SC(J,B)=SC(J,B)+
        SC(J,K):NEXT K:NEXT J
AP 410  FOR J=0 TO 1:FOR K=5
        TD B:Y$=STR$(SC(J,K))
        :L=LEN(Y$):TX=6+J*31-
        L
NP 420  TY=3+K:CX=TX-(TX<20):
        CY=TY:M$=Y$:GOSUB 11B
        0:NEXT K:NEXT J
GD 430  NEXT RR:REM ENO OF MA
        IN LOOP
MG 440  GET #1,TK:RUN
MM 450  PRINT "{CLEAR}";:PRIN
        T "{11 SPACES}{Q}{I}";
        :FOR J=1 TO 7:PRINT "
        {W}{I}";:NEXT J:PRINT
         "{E}":LL=7
BD 460  FOR J=0 TO 4:TX=9-2*J
        :TY=1+J*4:R$="{=} "
BC 470  FOR K=1 TO 2:CX=TX:CY
        =TY+K-1:M$="":GDSUB 1
        1B0
AL 480  PRINT "  ■ ";:GOSUB 5
        50:PRINT "■":NEXT K
GF 490  CX=TX:CY=TY+K-1:M$=""
        :GOSUB 11B0
PG 500  PRINT " {H}■ ";:GOSUB
         550:PRINT "■{J}":CX=
        TX:CY=TY+K:M$="":GOSU
        B 11B0
EF 510  LL=LL+2:PRINT "{H}
        {I}";:GOSUB 550:PRINT
         "{LEFT}{I}{J}";:NEXT
         J
BB 520  PRINT :PRINT "{RIGHT}
        {W} ";:GOSUB 550:PRIN
        T "{W}"
JP 530  R$="{U}{X}":LL=LL+1:P
        RINT "  {Z}";:GOSUB 55
        0:PRINT "{LEFT}{C}"
MJ 540  RETURN
MM 550  FOR L=1 TO LL:PRINT R
        $;:NEXT L:RETURN
CD 560  FOR J=1 TO 16:K=PT(RR
        ,J):JJ=2+J*2
LE 570  IF K>9 THEN L=INT(K/1
        0):T$=STR$(L):L$=T$(1
        ,1):GOTO 590
AB 580  L$=CHR$(32)
NE 590  CX=JJ:CY=22:M$=L$:GOS
        UB 11B0:CX=JJ:CY=23:T
        $=STR$(K):M$=T$(LEN(T
        $),LEN(T$)):GOSUB 11B
        0
DJ 600  NEXT J:RETURN
IG 610  FOR J=0 TO 4:SY=4+J*4
        :FOR K=0 TO J+3:SX=12
        -J*2+K*4
EE 620  CX=SX+1:CY=SY-1:M$="
        ":GOSUB 11B0
BB 630  M$=" ":CX=SX+1:CY=SY-
        1:GOSUB 11B0:WP=INT(R
        ND(1)*2)
PF 640  SW(J,K)=WP:SX(J,K)=0:
        GOSUB 760
DI 650  NEXT K:NEXT J
JD 660  FOR J=1 TO B:POKE SCR
        +50+J*2,16+J:NEXT J
JH 670  FOR J=0 TO 1:BX=J*31:
        CX=BX:CY=0:M$="":GOSU
        B 11B0
KK 680  PRINT "{B [illegible]}";
IN 690  FOR K=1 TO 5:CX=BX:CY
        =K:M$="":GDSUB 11B0:P
        RINT "{Y}{6 SPACES}
        {Y}";:NEXT K
BH 700  CX=BX:CY=K:M$="":GOSU
        B 11B0
KE 710  PRINT "{B [illegible]}";
FH 720  NEXT J:FOR TK=1 TO LE
        N(P1$):P1$(TK,TK)=CHR
        $(ASC(P1$(TK,TK))+12B
        ):NEXT TK
JM 725  FOR TK=1 TO LEN(P2$):
        P2$(TK,TK)=CHR$(ASC(P
        2$(TK,TK))+12B):NEXT
        TK
LO 730  CX=3-(LEN(P1$)=5):CY=
        0:M$=P1$:GOSUB 11B0
PF 740  CX=34-(LEN(P2$)=5):CY
        =0:M$=P2$:GOSUB 11B0
HM 750  RETURN
GL 760  CX=SX:CY=SY:M$=SP$(WP
        *3+1,WP*3+3):GOSUB 11
        B0:RETURN
BL 770  FOR J=0 TO 32:LB(J,0)
        =0:NEXT J:NB=1
KP 7B0  GET #1,A:A$=CHR$(A)
FB 790  IF A$="-" THEN RETURN
NH B00  IF A$="+" THEN A$=STR
        $(INT(RNO(1)*B+1))
NA B05  IF A$<"1" OR A$>"B" T
        HEN 7B0
LA B10  A=VAL(A$)
CA B20  LB(0,0)=1:FDR J=1 TO
        3:LB(0,J)=0:NEXT J:LB
        (0,4)=10+A*2
DI B30  EX=1:EV=0
CC B40  FOR J=0 TD 32:IF LB(J
        ,0) THEN EX=0:GOSUB B
        70
LK B50  NEXT J:SOUNO 1,0,0,0:
        SOUNO 3,0,0,0:SOUNO 2
        ,100,4,EV:EV=EV-(EV>0
        ):1F EX THEN RETURN
HC B60  GOTO B30
IG B70  DY=LB(J,0):DX=LB(J,1)
        :LY=LB(J,2):NY=LB(J,3
        ):NX=LB(J,4)
KB B80  SM=SCR+40+NX+LY*160+N
        Y*40:IF (LY+NY) THEN
        PDKE SM,0
JD B90  LB(J,3)=(NY+1)-4*(INT
        ((NY+1)/4)):ON NY+1 G
        DTO 900,920,960,970
PA 900  IF LY>4 THEN LB(J,0)=
        0:GOTO 1030:REM SCOR1
        NG ROUT1NE
KI 910  POKE SM+40,B4:DN INT(
        RNO(1)*3+1) GOTO 10B0
        ,1090,1100
HF 920  VX=0:GOSUB 1020:IF SX
        (WY,WX)=0 OR (SW(WY,W
        X)=SO)=0 THEN 940
EE 930  VX=1-2*SO:LB(J,1)=VX:
        LB(J,3)=NY+1:LB(J,4)=
        NX+VX:POKE SM+40+VX,B
        4:GOTO 1120
DM 940  IF SW(WY,WX)=SO THEN
        LB(J,0)=0:SX(WY,WX)=1
        :POKE SM+40,B4:GOTO 1
        110
KD 950  LB(J,3)=NY+1:POKE SM+
        40,B4:ON INT(RNO(1)*3
        +1) GOTO 10B0,1090,11
        00
CD 960  LB(J,1)=0:LB(J,4)=NX+
        DX:POKE SM+40+OX,B4:G
        OTO 1130
PD 970  LB(J,2)=LY+1:POKE SM+
        40,B4:GOSUB 1020:SW(W
        Y,WX)=1-SW(WY,WX)
BF 9B0  1F SX(WY,WX)=0 THEN 1
        010
AK 990  LB(NB,0)=1:LB(NB,1)=0
        :LB(NB,2)=LY:LB(NB,3)
        =0:LB(NB,4)=NX+2-SD*4
        :NB=NB+1
KK 1000 SX(WY,WX)=0:POKE SM-
        40+2-SO*4,0:GDSUB 11
        40
EA 1010 SX=12-WY*2+WX*4:SY=4
        +WY*4:WP=SW(WY,WX):G
        OSUB 760:GOTO 1120
EB 1020 WY=LY:JX=(NX/2)+LY-6
        :WX=INT(JX/2):SD=JX-
        2*(INT(JX/2)):RETURN
ME 1030 SF=PT(RR,NX/2-1)
LK 1040 SG=SC(QR,RR)+SF
JI 1050 TX=5+31*QR-(SG>9)-(S
        G>99)-(SG>999)
JF 1060 TY=1+RR:A$=STR$(SG)
IC 1070 CX=TX:CY=TY:M$=A$:GO
        SUB 11B0:SC(QR,RR)=S
        G:GOTD 1150
EJ 10B0 SOUNO 1,60,10,10:RET
        URN
HI 1090 SOUNO 1,121,10,10:RE
        TURN
EF 1100 SOUND 1,B1,10,10:RET
        URN
DI 1110 FOR A=10 TO 30:SOUNO
         1,A,12,10:NEXT A:SO
        UNO 1,0,0,0:RETURN
KE 1120 RETURN
IB 1130 FOR A=40 TO 20 STEP
```

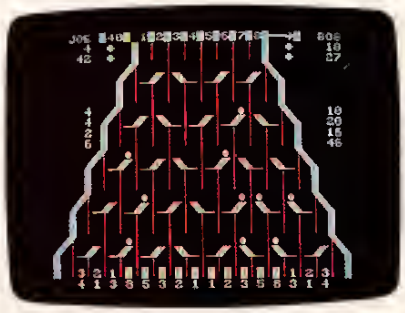
```
      -1:SOUND 1,A,12,10:N
      EXT A:SOUND 1,0,0,0:
      RETURN
MC 1140 SOUND 2,100,4,15:EV=
      15:RETURN
AA 1150 SOUND 1,121,10,10:SO
      UND 3,81,10,10
KI 1160 RETURN
MK 1170 REM CHAR COMMAND
CP 1180 POSITION CX,CY:PRINT
      M$;:RETURN
```

Program 4. Apple II Switchbox

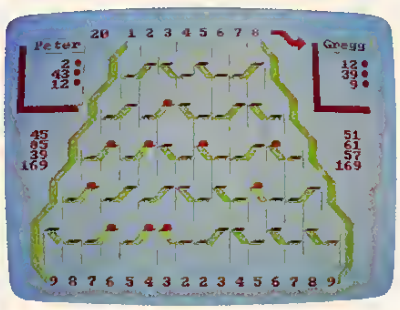
Version by Tim Victor, Editorial Programmer

```
4D 100 DIM SW(4,7,1),SP$(1),LB(3
      2,4),AR$(1),PT(4,16),SC(1
      ,8)
D4 110 SP$(0) = "%&)":SP$(1) = "
      )" + CHR$ (34):AR$(0) =
      "<--":AR$(1) = "-->":QR =
      1
D3 120 FOR J = 1 TO 4: READ PT(J
      ,0)
6C 130 FOR K = 1 TO 8: READ L:PT
      (J,K + 8) = L:PT(J,9 - K)
       = L: NEXT K,J
B4 140 DATA 10,2,2,2,2,2,2,2,2
AE 150 DATA 40,1,2,3,5,8,13,21,3
      4
B5 160 DATA 20,2,3,4,5,6,7,8,9
6E 170 DATA 80,1,4,9,16,25,36,49
      ,64
16 180 HOME : FLASH : PRINT "REA
      DING DATA STATEMENTS- ONE
       MOMENT": NORMAL
72 190 IF PEEK (768) < > 169 THE
      N POKE 230,64: GOSUB 970
6B 195 IF PEEK (190*256) = 76 TH
      EN PRINT CHR$ (4) "PR#A$3
      5C": GOTO 210
5C 200 POKE 54,92: POKE 55,3: CA
      LL 1002: POKE 6,0
E1 210 TEXT : HOME : INPUT "PLAY
      ER 1: ";P1$: INPUT "PLAYE
      R 2: ";P2$
32 220 PRINT "IS THIS CORRECT?":
       GET A$:A = ASC (A$): IF
      A < > 89 AND A < > 121 TH
      EN 210
EB 230 POKE 7,138: GOSUB 350: GO
      SUB 530
D8 240 FOR RR = 1 TO 4: POKE 7,1
      38: VTAB RR + 1: HTAB 8:
      PRINT "+";: HTAB 30: PRIN
      T "+";
58 250 GOSUB 490
2D 260 POKE 7,141:QR = 1 - QR: V
      TAB 1: HTAB 28 - QR * 21:
       PRINT " ";PT(RR,0);" ";:
       POKE 7,138: HTAB 8 + QR
      * 20: PRINT AR$(QR)
94 270 GOSUB 630: IF SC(1 - QR,R
      R) > = PT(RR,0) THEN 290
22 280 GOTO 260
69 290 POKE 7,141: FOR J = 0 TO
      1: FOR K = 5 TO 8:SC(J,K)
      = 0: NEXT K,J
56 300 FOR J = 0 TO 1: FOR K = 1
       TO 4:GL = PT(K,0):AC = S
      C(J,K):SC(J,5) = SC(J,5)
      + AC:SC(J,6) = SC(J,6) +
      (AC > = GL) * GL:SC(J,7)
      = SC(J,7) + (SC(J,K) - SC
      (1 - J,K)): NEXT K,J
4E 310 FOR J = 0 TO 1: FOR K = 6
       TO 7:SC(J,K) = SC(J,K) +
       SC(J,5): NEXT K,J
4B 320 FOR J = 0 TO 1: FOR K = 5
       TO 7:SC(J,8) = SC(J,8) +
       SC(J,K): NEXT K,J
4A 330 FOR J = 0 TO 1: FOR K = 5
       TO 8:Y$ = STR$ (SC(J,K))
      :L = LEN (Y$):TX = 6 + J
      * 31 - L:TY = 3 + K: VTAB
       TY: HTAB TX: PRINT Y$: N
      EXT K,J
C4 340 NEXT RR: VTAB 15: HTAB 16
      : PRINT "GAME OVER"
44 341 VTAB 17: HTAB 15: PRINT "
      PLAY AGAIN?"
44 345 GET G$: IF G$ = "" THEN 3
      45
54 346 IF G$ + "N" THEN : HOME :
       STOP
15 347 IF G$ = "Y" THEN RUN
C3 348 GOTO 345
E6 350 HOME : HGR2
CE 360 FOR I = 0 TO 5: FOR J = 0
       TO 1
BA 370 HTAB 11 - I * 2: VTAB I *
       4 + J + 2: PRINT "*";: H
      TAB 27 + I * 2: PRINT "*"
      : NEXT
2C 380 NEXT : FOR I = 0 TO 4
B4 390 HTAB 10 - I * 2: VTAB I *
       4 + 4: PRINT CHR$ (33);
      CHR$ (34);
97 400 HTAB 27 + I * 2: PRINT CH
      R$ (37); CHR$ (38);
B6 410 HTAB 9 - I * 2: VTAB I *
      4 + 5: PRINT CHR$ (35); C
      HR$ (36);
C6 420 HTAB 28 + I * 2: PRINT CH
      R$ (39); CHR$ (40)
02 430 NEXT
19 440 HCOLOR= 5: FOR I = 0 TO 6
      : FOR HP = 87 - I * 14 TO
       171 + I * 14 STEP 28
7E 450 VS = I * 32 - 28:VE = VS
      + 50: IF VS < 8 THEN VS =
       8
3D 460 IF VE > 182 THEN VE = 182
22 470 HPLOT HP,VS TO HP,VE: NEX
      T
D0 480 NEXT : RETURN
C8 490 POKE 7,141: FOR J = 1 TO
      16:K = PT(RR,J):JJ = 2 +
      J * 2
4D 500 IF K > 9 THEN L = INT (K
      / 10):L$ = STR$ (L): GOTO
       520
3E 510 L$ = " "
58 520 VTAB 23: HTAB JJ: PRINT L
      $: HTAB JJ: PRINT RIGHT$
      ( STR$ (K),1);: NEXT J: R
      ETURN
96 530 FOR J = 0 TO 4:SY = 5 + J
       * 4: FOR K = 0 TO J + 3:
      SX = 12 - J * 2 + K * 4
80 540 WP = INT ( RND (1) * 2)
15 550 SW(J,K,0) = WP:SW(J,K,1)
      = 0: GOSUB 620
31 560 NEXT K,J
A2 570 POKE 7,141
BB 580 VTAB 1: HTAB 12: FOR J =
      1 TO 8: PRINT J;" ";: NEX
      T
E0 590 VTAB 1: HTAB 3 - ( LEN (P
      1$) = 5): PRINT P1$;
FD 600 HTAB 34 - ( LEN (P2$) = 5
      ): PRINT P2$;
18 610 RETURN
27 620 VTAB 5Y: HTAB 5X: PRINT S
      P$(WP): RETURN
CB 630 FOR J = 0 TO 32:LB(J,0) =
       0: NEXT :N8 = 1
4B 640 GET A$: IF A$ = "-" THEN
      RETURN
FF 650 IF A$ = "+" THEN A$ = STR
      $ ( INT ( RND (1) * 8 + 1
      ))
2F 660 A = VAL (A$): IF A < 1 OR
       A > 8 THEN 640
F7 670 LB(0,0) = 1: FOR J = 1 TO
       3:LB(0,J) = 0: NEXT :LB(
      0,4) = 10 + A * 2
CA 680 EX = 1
B6 690 FOR J = 0 TO 32: IF LB(J,
      0) THEN EX = 0: GOSUB 720
2C 700 NEXT : IF EX = 0 THEN 680
19 710 RETURN
F3 720 OY = LB(J,0):OX = LB(J,1)
      :LY = LB(J,2):NY = LB(J,3
      ):NX = LB(J,4): IF (LY +
      NY) THEN GOSUB 1060
90 730 LB(J,3) = NY + 1 - (NY =
      3) * 4: ON NY + 1 GOTO 74
      0,760,790,800
E1 740 IF LY > 4 THEN LB(J,0) =
      0: GOTO 840
06 750 GOSUB 1080: ON INT ( RND
      (1) * 3 + 1) GOTO 880,890
      ,900
E4 760 VX = 0: GOSUB 830: IF SW(
      WY,WX,1) AND (SW(WY,WX,0)
       = SO) THEN VX = 1 - 2 *
      SO:LB(J,1) = VX:LB(J,3) =
       NY + 1:BX = NX + VX:LB(J
      ,4) = BX:BY = NY + LY * 4
       + 3: GOSUB 1090: GOTO 93
      0
7D 770 IF SW(WY,WX,0) = SO THEN
      LB(J,0) = 0:SW(WY,WX,1) =
       1: GOSUB 1080: GOTO 920
F1 780 LB(J,3) = NY + 1: GOSUB 1
      080: ON INT ( RND (1) * 3
       + 1) GOTO 880,890,900
64 790 LB(J,1) = 0:BX = NX + OX:
      LB(J,4) = BX:BY = NY + LY
       * 4 + 3: GOSUB 1090: GOT
      O 940
B4 800 LB(J,2) = LY + 1: GOSUB 1
      080: GOSUB 830:SW(WY,WX,0
      ) = 1 - SW(WY,WX,0)
62 810 IF SW(WY,WX,1) THEN LB(N8
      ,0) = 1:LB(NB,1) = 0:LB(N
      8,2) = LY:LB(NB,3) = 0:LB
      (NB,4) = NX + 2 - SO * 4:
      NB = N8 + 1:SW(WY,WX,1) =
       0:BX = NX + 2 - SO * 4:B
      Y = NY + LY * 4 + 1: GOSU
      B 1070: GOSUB 950
E9 820 SX = 12 - WY * 2 + WX * 4
      :SY = 5 + WY * 4:WP = BW(
      WY,WX,0): GOSUB 620: GOTO
       930
5D 830 WY = LY:JX = (NX / 2) + L
      Y - 6:WX = INT (JX / 2):S
      O = JX - INT (JX / 2) * 2
      : RETURN
E0 840 POKE 7,141:SF = PT(RR,NX
      / 2 - 1)
B6 850 SG = SC(QR,RR) + SF
4F 860 TX = 6 + 31 * QR - LEN (
      STR$ (SG))
DC 870 VTAB RR + 1: HTAB TX: PRI
      NT SG:SC(QR,RR) = SG: POK
      E 7,138: GOTO 960
05 880 POKE 776,80: GOTO 910
6A 890 POKE 776,160: GOTO 910
21 900 POKE 776,201: GOTO 910
69 910 POKE 781,200: POKE 841,1:
       POKE 849,196: POKE 798,9
      6: CALL 768: RETURN
44 920 POKE 776,208: POKE 781,22
      0: POKE 841,5: POKE 849,4
      : POKE 798,97: CALL 768:
      RETURN
03 930 POKE 776,232: POKE 781,25
      5: POKE 841,0: POKE 849,0
      : POKE 798,240: CALL 768:
       RETURN
FA 940 POKE 776,216: POKE 781,24
      0: POKE 841,4: POKE 849,4
      : POKE 798,240: CALL 768:
```

```
        RETURN
F8 950 PDKE 776,160: PDKE 781,16
       0: PDKE 841,1: PDKE 849,9
       6: PDKE 798,240: CALL 768
       : RETURN
E0 960 PDKE 776,160: PDKE 781,22
       0: PDKE 841,6: PDKE 849,6
       : PDKE 798,97: CALL 768:
       RETURN
00 970 FOR I = 768 TO 947: READ
       A: PDKE I,A: NEXT
AF 980 FOR I = 24576 TO 24831: P
       DKE I,128: NEXT
78 990 FOR I = 24832 TO 25087 ST
       EP 4: PDKE I,128: PDKE I
       + 1,136: PDKE I + 2,170:
       PDKE I + 3,136: NEXT
09 1000 FOR I = 35328 TO 35439:
       READ A: PDKE I,A: NEXT
51 1010 FOR I = 35552 TO 35559:
       READ A: PDKE I,A: NEXT
F5 1020 FOR I = 35568 TO 35575:
       READ A: PDKE I,A: NEXT
C0 1030 FOR I = 35704 TO 35711:
       READ A: PDKE I,A: NEXT
12 1040 FOR I = 36200 TO 36311:
       READ A: PDKE I,A: NEXT
00 1050 FOR I = 36360 TO 36599:
       READ A: PDKE I,A: NEXT :
        RETURN
04 1060 BY = NY + LY * 4 + 2:BX
       = NX
60 1070 VTAB BY: HTAB BX: PRINT
       " ": RETURN
C5 1080 BX = NX:BY = NY + LY * 4
       + 3
F0 1090 VTAB BY: HTAB BX: PRINT
       "D": RETURN
AF 1100 DATA 169,1,141,88,3,160,
       0,169
58 1110 DATA 160,141,49,3,169,25
       5,141,59
56 1120 DATA 3,173,59,3,141,90,3
       ,78
5A 1130 DATA 88,3,144,12,185,0,1
       45,200
64 1140 DATA 141,89,3,169,128,14
       1,88,3
FC 1150 DATA 78,89,3,144,3,173,4
       8,192
19 1160 DATA 162,0,232,208,253,1
       44,3,173
A3 1170 DATA 48,192,162,159,232,
       208,253,238
25 1180 DATA 90,3,208,211,24,173
       ,59,3
0C 1190 DATA 233,3,141,59,3,173,
       49,3
51 1200 DATA 105,3,141,49,3,144,
       186,96
6C 1210 DATA 8,5,0,255,216,120,1
       33,69
25 1220 DATA 134,70,132,71,166,7
       ,10,10
A0 1230 DATA 176,4,16,62,48,4,16
       ,1
44 1240 DATA 232,232,10,134,27,2
       4,101,6
50 1250 DATA 133,26,144,2,230,27
       ,165,40
F2 1260 DATA 133,8,165,41,41,3,5
       ,230
CC 1270 DATA 133,9,162,8,160,0,1
       77,26
95 1280 DATA 36,50,48,2,73,127,1
       64,36
39 1290 DATA 145,8,230,26,208,2,
       230,27
07 1300 DATA 165,9,24,105,4,133,
       9,202
50 1310 DATA 208,226,165,69,166,
       70,164,71
64 1320 DATA 88,76,240,253
5C 1330 DATA 0,0,0,0,0,0,0,0
98 1340 DATA 0,64,96,112,120,124
       ,126,127
E0 1350 DATA 127,63,31,15,7,3,1,
       0
E3 1360 DATA 64,96,112,120,124,1
       26,127,127
7D 1370 DATA 63,31,15,7,3,1,0,0
3F 1380 DATA 127,126,124,120,112
       ,96,64,0
A9 1390 DATA 0,1,3,7,15,31,63,12
       7
01 1400 DATA 126,124,120,112,96,
       64,0,0
44 1410 DATA 1,3,7,15,31,63,127,
       127
A0 1420 DATA 0,0,0,0,0,0,0,127
F7 1430 DATA 127,127,127,127,127
       ,127,127,127
EE 1440 DATA 0,0,28,62,127,62,28
       ,0
66 1450 DATA 0,0,0,0,0,0,0,0
7C 1460 DATA 0,0,0,127,127,0,0,0
7C 1470 DATA 0,12,6,127,127,6,12
       ,0
00 1480 DATA 0,24,48,127,127,48,
       24,0
9A 1490 DATA 0,0,28,62,62,62,28,
       0
0D 1500 DATA 0,0,0,0,14,0,0,0
FF 1510 DATA 0,0,0,14,0,14,0,0
F1 1520 DATA 0,60,102,48,24,0,24
       ,0
9C 1530 DATA 0,60,102,118,110,10
       2,60,0
C9 1540 DATA 0,24,28,24,24,24,60
       ,0
04 1550 DATA 0,60,102,48,12,102,
       126,0
29 1560 DATA 0,60,102,48,96,102,
       60,0
02 1570 DATA 0,48,56,52,126,48,4
       8,0
30 1580 DATA 0,126,6,62,96,102,6
       0,0
E1 1590 DATA 0,60,6,62,102,102,6
       0,0
A0 1600 DATA 0,126,96,48,24,12,1
       2,0
05 1610 DATA 0,60,102,60,102,102
       ,60,0
0F 1620 DATA 0,60,102,102,124,48
       ,24,0
06 1630 DATA 0,24,48,126,126,48,
       24,0
A0 1640 DATA 0,124,102,102,126,1
       02,102,0
0D 1650 DATA 0,62,102,102,62,102
       ,126,0
69 1660 DATA 0,60,102,6,6,102,62
       ,0
C0 1670 DATA 0,62,102,102,102,10
       2,62,0
0C 1680 DATA 0,126,6,6,62,6,126,
       0
46 1690 DATA 0,126,6,6,62,6,6,0
F0 1700 DATA 0,60,102,6,118,102,
       62,0
73 1710 DATA 0,102,102,102,126,1
       02,102,0
CC 1720 DATA 0,24,24,24,24,24,24
       ,0
77 1730 DATA 0,96,96,96,96,102,6
       0,0
64 1740 DATA 0,102,102,54,30,102
       ,102,0
DD 1750 DATA 0,6,6,6,6,6,126,0
07 1760 DATA 0,102,126,102,102,1
       02,102,0
0D 1770 DATA 0,62,102,102,102,10
       2,102,0
41 1780 DATA 0,60,102,102,102,10
       2,60,0
91 1790 DATA 0,62,102,102,62,6,6
       ,0
9C 1800 DATA 0,60,102,102,102,54
       ,108,0
46 1810 DATA 0,62,102,102,62,102
       ,102,0
11 1820 DATA 0,60,102,12,48,102,
       62,0
70 1830 DATA 0,126,24,24,24,24,2
       4,0
FD 1840 DATA 0,102,102,102,102,1
       02,62,0
02 1850 DATA 0,102,102,102,102,1
       02,24,0
04 1860 DATA 0,102,102,102,102,1
       26,102,0
1A 1870 DATA 0,102,102,102,60,10
       2,102,0
00 1880 DATA 0,102,102,102,60,24
       ,24,0
CE 1890 DATA 0,126,48,24,12,6,12
       6,0
5C 1900 DATA 0,0,0,0,0,0,0,0
60 1910 DATA 0,0,0,0,0,0,0,0
64 1920 DATA 0,0,0,0,0,0,0,0
6A 1930 DATA 0,0,24,60,60,24,0,0
```



Machine language creates the custom graphics in the Apple II version of "Switchbox."



IBM PC/PCjr "Switchbox," a colorful two-player game.

Program 5. IBM PC/PCjr Switchbox

Version by Tim Victor, Editorial Programmer

```
P0 100 RANDOMIZE TIMER
LO 110 SCREEN 1,0:CLS
LO 120 KEY OFF
6L 130 CDLOR 7,0
J0 140 DIM BOX(4,7,1),FALLING(32
       ,4),PDINTS(4,16),SCORE(1,
       8)
CA 150 DIM S(250),LEFTSW(35),RIG
       HTSW(35),BALL(4),UNBALL(4
       ),LARROW(35),RARROW(35)
QL 160 FOR J=1 TO 4:READ PDINTS(
       J,0)
CK 170 FOR K=1 TO 8:READ L:POINT
       S(J,K+8)=L:PDINTS(J,9-K)=
```

```
      L:NEXT K,J
GJ 180 DATA 10,2,2,2,2,2,2,2,2
GP 190 DATA 40,1,2,3,5,8,13,21,3
       4
DI 200 DATA 20,2,3,4,5,6,7,8,9
PM 210 DATA 80,1,4,9,16,25,36,49
       ,64
BD 220 LOCATE 1,1:INPUT "Player
       1: ";P1$:P1$=LEFT$(P1$,5)
DB 230 INPUT "Player 2: ";P2$:P2
       $=LEFT$(P2$,5)
KN 240 PRINT P1$;" vs ";P2$:PRIN
       T "Is this correct? (y/n)
       "
PP 250 YN$=INKEY$:IF YN$="" THEN
        250
DA 260 IF YN$="n" OR YN$="N" THE
       N 220
PJ 270 ROUND=4
PA 280 CLS:GOSUB 970:GOSUB 700
EG 290 CLS:GOSUB 1030:GOSUB 540
KP 300 PLAYER=1:PUT (225,1),RARR
       OW:FOR ROUND=1 TO 4:GOSUB
        500
PB 310 FOR I=0 TO 1:CIRCLE (53+2
       56*I,18+ROUND*8),2
DL 320 PAINT (53+256*I,18+ROUND*
       8),3:NEXT
EK 330 PLAYER=1-PLAYER:LOCATE 1,
       9+PLAYER*21:PRINT SPACE$(
       2)
QA 340 PUT (60,1),LARROW:PUT(225
       ,1),RARROW
CN 350 LOCATE 1,30-PLAYER*21:PRI
       NT RIGHT$(STR$(POINTS(ROU
       ND,0)),2)
DE 360 GOSUB 1210:IF SCORE(1-PLA
       YER,ROUND)<POINTS(ROUND,0
       ) THEN 330
KO 370 FOR J=0 TO 1:FOR K=5 TO 8
JI 380 SCORE(J,K)=0:NEXT K,J
DE 390 FOR J=0 TO 1:FOR K=1 TO 4
       :BONUS=POINTS(K,0):AMT=SC
       ORE(J,K)
HF 400 SCORE(J,5)=SCORE(J,5)+AMT
       :SCORE(J,6)=SCORE(J,6)-BO
       NUS*(AMT>=BONUS)
HL 410 SCORE(J,7)=SCORE(J,7)+SCO
       RE(J,K)-SCORE(1-J,K):NEXT
        K,J
AD 420 FOR J=0 TO 1:FOR K=6 TO 7
       :SCORE(J,K)=SCORE(J,K)+SC
       ORE(J,5):NEXT K,J
JA 430 FOR J=0 TO 1:FOR K=5 TO 7
       :SCORE(J,8)=SCORE(J,8)+SC
       ORE(J,K):NEXT K,J
EK 440 FOR J=0 TO 1:FOR K=5 TO 8
       :SCORE$=STR$(SCORE(J,K))
HA 450 LOCATE K+6,1+J*34:PRINT S
       PACE$(5)
NM 460 LOCATE K+6,5-LEN(SCORE$)+
       J*34:PRINT SCORE$:NEXT K,
       J
KS 470 NEXT ROUND:LOCATE 11,12:P
       RINT "Play again? (Y/N)"
PD 480 K$=INKEY$:IF K$="n" OR K$
       ="N" THEN CLS:END:ELSE IF
        K$="y" OR K$="Y" THEN RU
       N:ELSE GOTO 480
DD 500 FOR J=1 TO 16:K=POINTS(RO
       UND,J):JJ=3+J*2
IF 510 LOCATE 24,JJ:IF K>9 THEN
       PRINT MID$(STR$(K),2,1);
       ELSE PRINT " ";
BK 520 LOCATE 25,JJ:PRINT RIGHT$
       (STR$(K),1);:NEXT
MF 530 RETURN
QI 540 LINE (4,11)-(54,65),2,BF
NM 550 LINE (9,6)-(59,60),0,BF
IE 560 LINE (9,6)-(59,60),1,B
BA 570 LINE (10,18)-(58,18),1
EF 580 GET (4,6)-(59,65),S:PUT (
       262,6),S,PSET
NL 590 LOCATE 2,5-LEN(P1$)/2:PRI
       NT P1$
QP 600 LOCATE 2,37-LEN(P2$)/2:PR
       INT P2$
FD 610 FOR J=1 TO 8:LOCATE 1,J*2
       +10:PRINT J:NEXT
NC 620 FOR SWITCHY=0 TO 4
EH 630 FOR SWITCHX=0 TO SWITCHY+
       3
AC 640 WP=INT(RND(1)*2):BOX(SWIT
       CHY,SWITCHX,0)=WP:BOX(SWI
       TCHY,SWITCHX,1)=0
MC 650 GOSUB 670
JA 660 NEXT SWITCHX,SWITCHY:RETU
       RN
BI 670 SY=24+SWITCHY*32:SX=92-SW
       ITCHY*16+SWITCHX*32
IO 680 IF WP=0 THEN PUT (SX,SY),
       LEFTSW,PSET ELSE PUT (SX,
       SY),RIGHTSW,PSET
NC 690 RETURN
PP 700 FOR I=1 TO 10:LINE (I+155
       ,52)-(I+157,51),2
BF 710 LINE (I+189,20)-(I+191,19
       ),2:NEXT
BO 720 FOR I=172 TO 180:LINE (I,
       12)-(I+10,22),1
MD 730 LINE (I+1,11)-(I+3,10),2
LF 740 LINE (I,44)-(I-10,54),1
ID 750 LINE (I+1,43)-(I+3,42),2:
       NEXT
HD 760 LINE (186,21)-(200,22),1,
       BF
NB 770 LINE (166,53)-(154,54),1,
       BF
BD 780 GET (172,6)-(202,22),LEFT
       SW
MF 790 GET (154,38)-(184,54),RIG
       HTSW
LE 800 ARC=3.14159/2
ML 810 FOR I=1 TO 2:CIRCLE (80,8
       ),I*4,3,ARC,ARC*2
GD 820 CIRCLE (68,6),I*4,3,ARC*3
       ,ARC*4
DC 830 CIRCLE (231,8),I*4,3,0,AR
       C
DP 840 CIRCLE (243,6),I*4,3,ARC*
       2,ARC*3:NEXT
MP 850 LINE (80,1)-(88,4),3,BF
PI 860 LINE (231,1)-(223,4),3,BF
FB 870 LINE (61,9)-(67,13),3,BF
ML 880 LINE (250,9)-(244,13),3,B
       F
BG 890 PAINT (74,7),3
LJ 900 PAINT (237,7),3
HD 910 FOR I=5 TO 17:LINE (58,11
       )-(64,I),3
EM 920 LINE (253,11)-(247,I),3:N
       EXT
IL 930 GET (58,1)-(88,17),LARROW
EP 940 GET (223,1)-(253,17),RARR
       OW
NN 950 RETURN
KH 960 '--------- DRAW SWITCHBOX
        ---------
IM 970 CIRCLE (100,100),3,3
GE 980 PAINT (100,100),3
NE 990 GET (97,97)-(103,103),BAL
       L
AH 1000 PUT (97,97),BALL,PRESET
HL 1010 GET (97,97)-(103,103),UN
       BALL
IB 1020 RETURN
BB 1030 LINE (80,24)-(87,39),1,B
       F
MC 1040 LINE (224,24)-(231,39),1
       ,BF
MH 1050 FOR I=0 TO 7:LINE (81+I,
       23)-(96+I,8),1
PM 1060 LINE (208+I,8)-(223+I,23
       ),1:NEXT
PD 1070 GET (80,8)-(103,39),S
HP 1080 FOR I=0 TO 3:PUT (64-I*1
       6,I*32+40),S:NEXT
HK 1090 GET (208,8)-(231,39),S
JE 1100 FOR I=0 TO 3:PUT (224+I*
       16,I*32+40),S:NEXT
LK 1110 LINE (96,8)-(215,15),0,B
       F
DL 1120 LINE (16,168)-(23,183),1
       ,BF
KI 1130 LINE (288,168)-(295,183)
       ,1,BF
NG 1140 FOR I=0 TO 7:LINE (16+I,
       184)-(29,197-I),1
KK 1150 LINE (288+I,184)-(282,19
       0+I),1:NEXT
BF 1160 FOR I=0 TO 6:FOR HP=123-
       I*16 TO 187+I*16 STEP 32
MF 1170 VS=I*32-32:VE=V5+64:IF V
       5<8 THEN VS=8
ME 1180 IF VE>186 THEN VE=186
DK 1190 LINE (HP,VS)-(HP,VE),1:N
       EXT:NEXT
IP 1200 RETURN
IK 1210 'GAME STUFF
FD 1220 FOR F8ALL=0 TO 32:FALLIN
       G(F8ALL,0)=0:NEXT:NEW8AL
       L=1
LA 1230 A$="":WHILE A$="":A$=INK
       EY$:WEND
GE 1240 IF A$="-" THEN RETURN
PK 1250 IF A$="+" THEN A$=CHR$(I
       NT(RND(1)*8+49))
EI 1260 A=VAL(A$):IF A<1 OR A>8
       THEN 1230
HK 1270 FALLING(0,0)=1:FOR J=1 T
       O 3:FALLING(0,J)=0:NEXT
BB 1280 FALLING(0,4)=10+A*2
QL 1290 EXIT=0:WHILE EXIT=0:EXIT
       =1
DH 1300 FOR F8ALL=0 TO 32:IF FAL
       LING(F8ALL,0)=1 THEN EXI
       T=0:GOSUB 1320
AA 1310 NEXT:WEND:RETURN
FM 1320 OY=FALLING(F8ALL,0):OX=F
       ALLING(F8ALL,1):LEVEL=FA
       LLING(F8ALL,2)
HA 1330 NY=FALLING(F8ALL,3):NX=F
       ALLING(F8ALL,4)
IH 1340 IF LEVEL<>0 OR NY<>0 THE
       N GOSUB 1570
AB 1350 NY=NY+1:FALLING(F8ALL,3)
       =NY AND 3:ON NY GOTO 136
       0,1380,1420,1430
FH 1360 IF LEVEL=5 THEN FALLING(
       F8ALL,0)=0:GOTO 1500
BM 1370 GOSUB 1560:ON INT(RND(1)
       *3+1) GOTO 1580,1590,160
       0
QK 1380 VX=0:GOSUB 1540
LC 1390 IF BOX(SWITCHY,SWITCHX,1
       )=1 AND BOX(SWITCHY,SWIT
       CHX,0)=SIDE THEN VX=1-2*
       SIDE:FALLING(F8ALL,1)=VX
       :NX=NX+VX:FALLING(F8ALL,
       4)=NX:GOSUB 1560:GOTO 16
       10
DM 1400 GOSUB 1560:IF BOX(SWITCH
       Y,SWITCHX,0)=SIDE THEN F
       ALLING(F8ALL,0)=0:BOX(SW
       ITCHY,SWITCHX,1)=1:GOTO
       1620
PA 1410 ON INT(RND(1)*3+1) GOTO
       1580,1590,1600
IG 1420 FALLING(F8ALL,1)=0:NX=NX
       +OX:FALLING(F8ALL,4)=NX:
       GOSUB 1560:GOTO 1630
PE 1430 FALLING(F8ALL,2)=LEVEL+1
       :GOSUB 1560
CL 1440 GOSUB 1540:BOX(SWITCHY,S
       WITCHX,0)=1-BOX(SWITCHY,
       SWITCHX,0)
DB 1450 IF BOX(SWITCHY,SWITCHX,1
       )=0 THEN 1490
QI 1460 FALLING(NEW8ALL,0)=1:FAL
       LING(NEW8ALL,1)=0:FALLIN
       G(NEW8ALL,2)=LEVEL
BL 1470 FALLING(NEW8ALL,3)=0:FAL
```

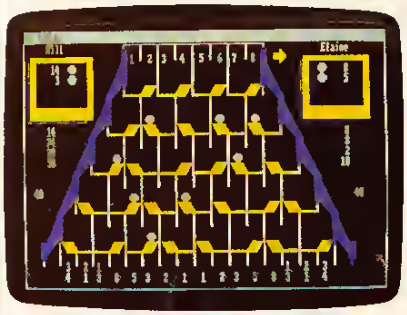
```
        LINE(NEWBALL,4)=NX+2-SIZE*4
MD 1480 BOX(SWITCHY,SWITCHX,1)=0:NEWBALL=NEWBALL+1:GOSUB 1640
OO 1490 WP=BOX(SWITCHY,SWITCHX,0):GOSUB 670:GOTO 1630
PD 1500 AMT=POINTS(ROUNO,NX/2-1):SUBTOT=SCORE(PLAYER,ROUNO)+AMT
NJ 1510 SUB$=STR$(SUBTOT):LOCATE ROUNO+3,7-LEN(SUB$)+PLAYER*32:PRINT SUB$
MO 1520 SCORE(PLAYER,ROUNO)=SUBTOT
AC 1530 GOTO 1650
LM 1540 SWITCHY=LEVEL:JX=NX/2+LEVEL-6
JL 1550 SWITCHX=INT(JX/2):SIOE=JX-INT(JX/2)*2:RETURN
CO 1560 PUT (NX*8,8+LEVEL*32+NY*8),BALL,OR:RETURN
OE 1570 PUT (NX*8,8+LEVEL*32+NY*8),UNBALL,ANO:RETURN
QO 1580 FOR I=0 TO 1:SOUNO 880,1:SOUNO 32767,1:NEXT:RETURN
KL 1590 FOR I=0 TO 1:SOUNO 660,1:SOUNO 32767,1:NEXT:RETURN
CM 1600 FOR I=0 TO 1:SOUNO 440,1:SOUNO 32767,1:NEXT:RETURN
OO 1610 FOR I=1 TO 6:SOUNO 1100*RNO(1)+37,1:NEXT:RETURN
IM 1620 FOR I=800 TO 200 STEP -20:SOUNO I,.1:NEXT:RETURN
GA 1630 FOR I=1 TO 6:SOUNO 550*RNO(1)+37,1:NEXT:RETURN
FB 1640 FOR I=0 TO 1:FOR J=440 TO 880 STEP 80:SOUNO J,.5:NEXT J,I:RETURN
KJ 1650 FOR I=0 TO 5:SOUNO 330,.5:SOUNO 440,.5:SOUNO 550,.5:NEXT
KI 1660 SOUNO 32767,1:RETURN
```

Program 6. Amiga Switchbox

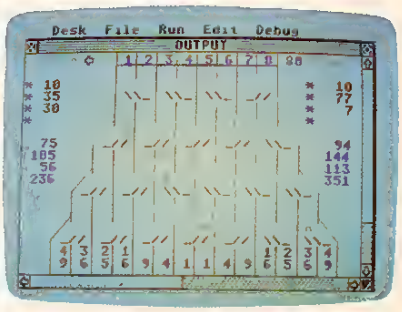
Version by Philip I. Nelson, Assistant Editor

```
'Switchbox for 512K Amiga←
'Set Preferencee for 80 columns←

Reetart:←
CLEAR:GOSUB Setup←

Main:←
FOR Round=1 TO 4←
PUT (80,7+Round*6),Ball←
PUT (818,7+Round*6),Ball←
GOSUB Valuee←
SAY TRANSLATE$(Intro$(Round))←
Keepgoing:←
Who=1-Who 'alternate playsrs←
GOSUB Taksturn←
IF SC(1-Who,Round)=>Pointe(Round,0) THEN Nextround←
GOTO Ksepgoing←

Nextround:←
FOR j=0 TO 1:FOR k=5 TO 6←
SC(j,k)=0:NEXT:NEXT←
FOR j=0 TO 1:FOR k=1 TO 4←
gx=Points(k,0) ac=SC(j,k)←
SC(j,5)=SC(j,5)+ac←
SC(j,5)=SC(j,5)-(ac=>gx)*gx←
SC(j,7)=SC(j,7)+SC(j,k)-SC(1-j,k)←
NEXT:NEXT←
FOR j=0 TO 1:FOR k=5 TO 7←
SC(j,k)=SC(j,k)+SC(j,5)←
NEXT:NEXT←
FOR j=0 TO 1:FOR k=5 TO 7←
SC(j,8)=SC(j,5)+SC(j,k)←
NEXT:NEXT←
FOR j=0 TO 1←
FOR k=5 TO 5:y$=STR$(SC(j,k))←
x=LEN(y$):tx=5+j*64-x:ty=4+k←
LOCATE ty,tx-1:PRINT SPACE$(2)←
LOCATE ty,tx:PRINT y$←
NEXT NEXT←
NEXT Round←

Gohome:←
LINE (240,70)-(352,100),2,bf←
LOCATE 11,32:PRINT " Play again? "←
text$=Who$(ABS(SC(1,8)>SC(0,5)))←
tsxt$=text$+" wine this game.."←
tsxt$=text$+"How about anothsr?"←
SAY TRANSLATE$(tsxt$),Voice%←
FOR j=0 TO 10:x$=INKEY$:NEXT←
Again:←
x$=INKEY$:IF x$="" THEN Again←
SAY TRANSLATE$("OK."),Voice%←
IF x$="y" OR x$="Y" THEN WINDOW CLOSE 2:GOTO Reetart←
SAY TRANSLATE$(" Bye-bye."),Voice%←
WINDOW CLOSE 2←
END←

Taketurn:←
FOR j=0 TO nb:LB(j,0)=0:NEXT:nb=1←
SAY TRANSLATE$(Who$(Who)+CHR$(45))←
PUT (140,5),Larrow:PUT (440,5),Rarrow←
FOR j=0 TO 9:x$=INKEY$:NEXT←
Getksy:←
a$=INKEY$:IF a$="-" THEN RETURN←
IF a$="+" THEN a$=STR$(INT(RND(1)*5+1))←
a=VAL(a$):IF (a<1) OR (a>8) THEN Gstkey←
LB(0,0)=1←
FOR j=1 TO 3:LB(0,j)=0:NEXT←
LB(0,4)=a+3←
Moreballe:←
ex=1:FOR j=0 TO nb←
IF LB(j,0) THEN ex=0:GOSUB Movsone←
NEXT:IF ex=0 THEN Moreballe←
x=0:FOR j=13 TO 7 STEP -3:FOR k=x TO 15-x←
PUT (Column(k),Row(j)+1),Blank,AND←
NEXT:x=x+1:NEXT:RETURN←

Movsone:←
dy=LB(j,0):dx=LB(j,1):LY=LB(j,2)←
ny=LB(j,3):nx=LB(j,4)←
IF ny THEN←
PUT (Column(nx),Row(ny+(LY*3))+1),Blank,AND←
END IF←
LB(j,3)=(ny+1) MOD 3←
ON ny+1 GOTO Pos0,Poe1,Poe2←

Poe0:←
IF LY>4 THEN LB(j,0)=0:GOTO Score←
vx=0:GOSUB Whichway←
IF (SW(wx,wy,1)) AND (SW(wx,wy,0)=sd) THEN←
vx=1-2*ed:LB(j,3)=ny+1:LB(j,4)=nx+vx←
GOTO Putball←
END IF←
IF SW(wx,wy,0)=ed THEN←
LB(j,0)=0 ←
SW(wx,wy,1)=1:ny=ny+1←
GOTO Putball←
END IF←
LB(j,3)=ny+1:GOTO Putball←

Pos1:←
LB(j,1)=0:LB(j,4)=nx+dx:GOTO Putbal
1←
←
Poe2:←
LB(j,2)=LY+1:GOSUB Whichway←
SW(wx,wy,0)=1-SW(wx,wy,0) ←
IF SW(wx,wy,1) THEN←
PUT (Column(LB(j,4)+1-ed*2),Row(ny+(LY*3))),Blank,AND←
LB(nb,0)=1:LB(nb,1)=0:LB(nb,2)=LY←
LB(nb,3)=0:LB(nb,4)=nx+1-ed*2:nb=nb+1←
SW(wx,wy,1)=0←
END IF←
ex=Xpos(wx,wy):sy=Ypos(wx,wy)←
wp=SW(wx,wy,0)←
'Always fall thru to ewitch←

Switch:←
PUT (sx,ey),Swblank, AND←
ON wp+1 GOTO Lsft,Right←
Left:←
PUT (sx,sy),Lewitch,OR:GOTO Bop←
Right:←
PUT (ex,ey),Rswitch,OR←
Bop:←
SOUND 100,1,64,Who←
SOUND 250,1,64,3-Who←
RETURN←

Putball:←
SOUND INT(RND(1)*10)*(30*LY)+200,1,64,Who←
PUT (Column(nx),Row(ny+(LY*3)+1)),Ball,OR←
RETURN←

Whichway:←
wx=LY:wy=INT((nx+LY-4)/2):sd=(nx+LY) AND 1:RETURN←

Score:←
ef=Points(Round,nx+1):eg=SC(Who,Round)+ef←
tx=8+83*Who+(eg>9)+(eg>99)+(sg>999)←
ty=2+Round:a$=MID$(STR$(sg),2)←
LOCATE ty,tx.PRINT a$←
SC(Who,Round)=sg←
FOR j=1500 TO 200 STEP -300←
SOUND j,1,64,Who←
SOUND j+400,1,64,3-Who←
NEXT:RETURN←

Valuss:←
FOR j=0 TO 1←
k=2+70*j:LOCATE 15,k←
PRINT SPACE$(3):LOCATE 15,k←
PRINT RIGHT$(STR$(Pointe(Round,0)),3)←
NEXT←
FOR j=1 TO 15:k=Pointe(Round,j)←
m=5+j*3.75←
IF k>9 THEN←
x=INT(k/10)←
x$=MID$(STR$(x),2,1)←
ELSE←
x$=CHR$(32)←
END IF←
LOCATE 22,m:PRINT x$;←
```

```
LOCATE 23,m:PRINT RIGHT$(STR$(k),1
);←
NEXT:RETURN←

Sstup:←
RANDOMIZE TIMER←
DIM Voics%(6)
FOR j=0 TO 6
READ Voice%(j):NEXT
DATA 110,0,160,0,22200,64,10,1,0
Grsst$="Hi. Welcome to Switchbox."←
PRINT Greet$
SAY TRANSLATE$(Greet$),Voice%←
SCREEN 2,640,200,2,2←
PALETTE 0, 0, 0, 0 ←
PALETTE 1, 1, 1, 1←
PALETTE 2, 0, .1, .7 ←
PALETTE 3, 1, 1, .13←
WINDOW 2,"Switchbox",,0←
DIM Larrow(30),Rarrow(30),Wav%(256
),Lefthunk(400)←
DIM Righthunk(400),Swblank(100),Rs
witch(200)←
DIM Lewitch(200),Column(16),Row(25)
←
DIM Blank(70),Ball(80),Piecs(80)←
DIM SW(6,6,1),LB(32,4),Pointe(4,16),SC
(1,6)←
FOR j=0 TO 10:LINE (0,5)-(10,j),3←
NEXT←
LINE (10,3)-(20,7),3,bf←
GET (0,0)-(20,10),Larrow←
PUT (0,0),Larrow←
FOR j=0 TO 10←
LINE (20,6)-(10,j),3←
NEXT←
LINE (0,3)-(10,7),3,bf←
GET (0,0)-(20,10),Rarrow←
PUT (0,0),Rarrow←
GET (6,2)-(22,9),Blank←
CIRCLE (15,4),7,1←
PAINT (16,4),1←
GET (6,0)-(22,9),Ball←
PUT (8,0),Ball←
FOR j=0 TO 127:Wav%(j)=-127←
Wav%(j+126)=127:NEXT←
FOR j=0 TO 3:WAVE j,Wav%←
NEXT←
DATA 10,"round 1. equal scores."←
DATA 2,2,2,2,2,2,2,2←
DATA 40,"round 2. fibonachie seequenc
e."←
DATA 1,2,3,6,6,13,21,34←
DATA 20,"round 3. arithmetic seequsn
cs."←
DATA 2,3,4,5,6,7,8,9←
DATA 60,"round 4. seequence of square
s."←
DATA 1,4,9,16,25,36,49,64←
FOR j=1 TO 4:READ Pointe(j,0)←
READ Intro$(j)←
FOR k=1 TO 6:READ x←
Points(j,k+6)=x:Pointe(j,9-k)=x←
NEXT k:NEXT j←
a=218:b=2←
FOR j=0 TO 4←
a=a-30:b=b+30←
FOR k=0 TO j+3←
Xpos(j,k)=a+k*60←
Ypos(j,k)=b←
NEXT:NEXT←
k=0←
FOR j=70 TO 520 STEP 30←
Column(k)=j←
k=k+1:NEXT←
k=0←
FOR j=4 TO 164 STEP 10←
Row(k)=j:k=k+1:NEXT←

Start:←
SAY TRANSLATE$("First player's nams
?"),Voice%←
INPUT"Name of Playsr 1";p0$←
SAY TRANSLATE$("Second player'e na
me?"),Voice%←
INPUT"Name of Playsr 2";p1$←
Who$(0)=LEFT$(p0$,6):Who$(1)=LEF
T$(p1$,6)←
text$=Who$(0)+" plays "+Who$(1)+"
. Is this correct"←
PRINT text$;
SAY TRANSLATE$(text$),Voice%←
INPUT qusry$:an$=LEFT$(query$,1)←
IF LEN(an$)=0 OR an$="y" OR an$="
Y" THEN Draw←
GOTO etart←

Draw:←
SAY TRANSLATE$("OK."),Voice%←
CLS←
LOCATE 1,6:PRINT Who$(0)←
LOCATE 1,66:PRINT Who$(1)←
x=4:FOR j=0 TO 1 'score boxes←
LINE (x,12)-(x+110,60),2,bf'shadow←
LINE (x+8,10)-(x+120,66),3,bf'outli
ns←
LINE (x+16,14)-(x+110,46),0,bf'insi
ds←
x=x+460:NEXT ←
x=1:FOR j=24 TO 60 STEP 3.7←
LOCATE 2,j:PRINT x←
x=x+1:NEXT←
LINE (160,0)-(162,40),,bf←
GET (160,0)-(182,40),Piece←
LINE (160,0)-(420,0)←
FOR j=210 TO 420 STEP 80←
LINE (j,0)-(j+2,12),,bf←
PUT (j,40),Piece←
PUT (j,100),Piece←
NEXT←
FOR j=160 TO 420 STEP 80←
PUT (j,0),Piscs,OR←
PUT (j,70),Piece←
PUT (j,126),Piece←
NEXT←
PUT (120,126),Pisce←
PUT (150,100),Piece←
PUT (450,100),Piecs←
PUT (480,126),Piscs←
ERASE Piece 'rsclaim msmory←
FOR j=30 TO 570 STEP 30←
LINE (j,156)-(j+2,170),1,bf←
NEXT←
LINE (176,4)-(166,32),2,bf←
LINE (416,4)-(426,32),2,bf←
LINE (176,32)-(156,42),2←
LINE STEP(0,0)-STEP(-10,0),2←
LINE STEP(0,0)-STEP(35,-32),2←
PAINT (176,31),2←
LINE (426,32)-(446,42),2←
LINE STEP(0,0)-STEP(10,0),2←
LINE STEP(0,0)-STEP(-36,-32),2←
PAINT (427,32),2←
GET (136,12)-(166,69),Lefthunk←
GET (416,12)-(466,62),Righthunk←
l=106:r=446:k=42←
FOR j=1 TO 4←
PUT (l,k),Lsfthunk,OR←
PUT (r,k),Righthunk,OR←
l=l-30:r=r+30:k=k+30←
NEXT←
ERASE Lefthunk,Righthunk←
LINE (26,163)-(36,166),2,bf←
LINE (564,163)-(576,166),2,bf←
GET (245,32)-(299,40),Swblank←
FOR j=0 TO 16←
LINE (270+j,40)-(260+j,32),3←
NEXT←
LINE (246,39)-(260,40),3,bf←
GET (245,32)-(296,40),Rswitch←
PUT (164,32),Swblank,AND←
FOR j=0 TO 20←
LINE (164+j,32)-(193+j,40),3←
NEXT←
LINE (193,39)-(236,40),3,bf←
GET (164,32)-(236,40),Lswitch←
FOR m=0 TO 4:FOR n=0 TO m+3←
ex=Xpos(m,n):sy=Ypoe(m,n)←
wp=INT(RND(1)*2)←
SW(m,n,0)=wp←
SW(m,n,1)=0←
Who=1-Who:GOSUB Switch←
NEXT n:NEXT m←
PUT (140,6),Larrow←
RETURN←
```



The Amiga version of "Switchbox" features speech and stereo sound effects.



"Switchbox" for the Atari 520ST computer.

Program 7. Atari 520ST Switchbox

Version by Kevin Mykytyn, Editorial Programmer

```
10 restore:dim sw(4,7,1),sp$(1),lb(32,4),ar$(
    1),pt(4,16),sc(1,8):qr=1
20 sp$(0)=" \ \_:sp$(1)="_//":ar$(0)=C
    HR$(4)+" ":ar$(1)=" "+CHR$(3)
30 color 1,1,1,1,1:Q1=-2:Q2=0:FOR J=
    1 to 4:read pt(j,0)
40 for a=0 to 1:for b=0 to 8:sc(a,b)=0:ne
    xt:next
50 for k=1 to 7:read l:pt(j,k+7)=l:pt(j,8-
    k)=l:next k,j
60 data 10
70 data 2,2,2,2,2,2,2
80 data 40
90 data 1,2,3,5,8,13,21
100 data 20
110 data 2,3,4,5,6,7,8
120 data 80
130 data 1,4,9,16,25,36,49
```

```
140 fullw 2:clearw 2:gotoxy 0,0:input "PLAYER 1";p1$
150 input "PLAYER 2";p2$:p1$=left$(p1$,5):p2$=left$(p2$,5):print p1$;" VS ";p2$
160 print "IS THIS CORRECT?":GK=INP(2):if gk<>asc("Y") and gk<>asc("y") then 140
170 gosub 410:gosub 510:color 1,1,1
180 for rr=1 to 4:color 5:gotoxy 0,1+rr:print "*";:gotoxy 28,1+rr:print "*"
190 gosub 450:rem put scores at bottom
200 qr=1-qr:ty=qr*20:tx=26-ty:cx=tx:cy=0
210 color 5:m$=right$(str$(pt(rr,0)),2):gosub 1110
220 CX=6+ty:cy=0:m$=ar$(qr):gosub 1110
230 gosub 660:if sc(1-qr,rr) >= pt(rr,0) then 250:rem end of round
240 goto 200
250 for j=0 to 1:for k=5 to 8:sc(j,k)=0:next k,j
260 for j=0 to 1:for k=1 to 4:gl=pt(k,0):ac=sc(j,k):sc(j,5)=sc(j,5)+ac
270 sc(j,6)=sc(j,6)-(ac>=gl)*gl:sc(j,7)=sc(j,7)+(sc(j,k)-sc(1-j,k)):next k,j
280 for j=0 to 1:for k=6 to 7:sc(j,k)=sc(j,k)+sc(j,5):next k,j
290 for j=0 to 1:for k=5 to 7:sc(j,8)=sc(j,8)+sc(j,k):next k,j
300 for j=0 to 1:for k=5 to 8:y$=str$(sc(j,k)):l=len(y$):tx=5+j*28-1
310 ty=2+k:cx=tx+(tx<20):cy=ty:m$=y$:color 4:gosub 1110:next k,j
320 next rr:rem end of main loop
330 color 1,1,8
340 gotoxy 9,10:print spc (19)
350 gotoxy 9,11:print " PLAY AGAIN? (Y/N) "
360 gotoxy 9,12:print spc (19)
370 for a=78 to 231 step 153:linef a,100,a,109:next
380 linef 78,100,231,100:linef 78,109,231,109
390 a=inp(2):if a=asc("Y") or a=asc("y") then clear:goto 10 else end
410 clearw 2:color 4,1,6
420 for j=1 to 8:gotoxy 7+2*j,0:print j:next
430 for j=82 to 227 step 18.125:linef j,0,j,190:next
440 linef 82,9,227,9:return
450 for j=1 to 14:k=pt(rr,j):jj=2+j*2
460 if k>9 then l=int(k/10):l$=mid$(str$(l),2,1):goto 480
470 l$=chr$(32)
480 gotoxy jj,16:print l$;:cx=jj:cy=17
490 gotoxy jj,17:print right$(str$(k),1);
500 next j:return
510 gosub 580:for j=0 to 3:sy=4+j*4:for k=0 to j+3:sx=12-j*2+k*4
520 cx=sx-1:cy=sy-2:m$=" ":gosub 1110:color 0,0,0,0,0
530 linef cx*9,cy*9+10,cx*9+3,cy*9+10:wp=int(rnd(1)*2)
540 sw(j,k,0)=wp:sw(j,k,1)=0:gosub 650
550 next k,j:color 11
560 cx=1:cy=0:m$=p1$:gosub 1110
570 cx=29:cy=0:m$=p2$:gosub 1110:return
580 linef 82,51,64,69:linef 64,69,64,172
590 linef 64,87,46,105:linef 46,105,46,172
600 linef 46,123,28,141:linef 28,141,28,172
610 linef 228,51,246,69:linef 246,69,246,172
620 linef 246,87,264,105:linef 264,105,264,172
630 linef 264,123,282,141:linef 282,141,282,172
640 return
650 color 2:cx=sx-2:cy=sy-1:m$=sp$(wp):gosub 1110:return
660 for j=0 to 32:lb(j,0)=0:next:nb=1
670 a=inp(2):a$=chr$(a)
680 if a$="-"then return
690 if a$="+"then a$=str$(int(rnd(1)*8+1))
700 a=val(a$):if (a<1)or(a>8) then 670
710 lb(0,0)=1:for j=1 to 3:lb(0,j)=0:next:lb(0,4)=10+a*2
720 ex=1
730 for j=0 to 32:if lb(j,0) then ex=0:gosub 760
740 next:sound 1,0,0,0:if ex then return
750 goto 720
760 dy=lb(j,0):dx=lb(j,1):ly=lb(j,2):ny=lb(j,3):nx=lb(j,4):QT=LY*4+NY
770 if (ly+ny) AND LY<4 then gotoxy nx+q1,ly*4+ny+q2:print " "
780 color 11:lb(j,3)=(ny+1) and 3:on ny+1 goto 790,810,860,880
790 if ly>3 then lb(j,0)=0:goto 950:rem scoring routine
800 gosub 1120:on int(rnd(1)*3+1) goto 1000,1010,1020
810 vx=0:gosub 940:if sw(wy,wx,1)=0 or (sw(wy,wx,0)=sd)=0 then 840
820 vx=1-2*sd:lb(j,1)=vx:lb(j,3)=ny+1
830 lb(j,4)=nx+vx:gotoxy nx+q1+vx,ly*4+ny+q2-(qt<15):print "o":goto 1050
840 if sw(wy,wx,0)=sd then lb(j,0)=0:sw(wy,wx,1)=1:gosub 1120:goto 1030
850 lb(j,3)=ny+1:gosub 1120:on int(rnd(1)*3+1) goto 1000,1010,1020
860 lb(j,1)=0:lb(j,4)=nx+dx:gotoxy nx+q1+dx,ly*4+ny+q2-(qt<15):print "o"
870 goto 1060
880 if qt<15 then gosub 1120
890 lb(j,2)=ly+1:gosub 940:sw(wy,wx,0)=1-sw(wy,wx,0)
900 if sw(wy,wx,1)=0 then 930
910 lb(nb,0)=1:lb(nb,1)=0:lb(nb,2)=ly:lb(nb,3) =0:lb(nb,4)=nx+2-sd*4:nb=nb+1
920 sw(wy,wx,1)=0:gotoxy nx+q1+2-sd*4,ly*4+ny+q2-1:print"":gosub 1070
930 sx=12-wy*2+wx*4:sy=4+wy*4:wp=sw(wy,wx,0):gosub 650:goto 1050
940 wy=ly:jx=int(nx/2)+ly-6:wx=int(jx/2):sd=jx and 1:return
950 sf=pt(rr,nx/2-2)
960 sg=sc(qr,rr)+sf
970 tx=3+29*qr+(sg>9)+(sg>99)+(sg>999)
980 ty=rr+1:a$=mid$(str$(sg),2):color 6
990 cx=tx:cy=ty:m$=a$:gosub 1110:sc(qr,rr)=sg:goto 1080
1000 sound 1,15,1,3:wave 1,1,12,90,0:return
1010 sound 1,15,1,4:wave 1,1,12,90,0:return
1020 sound 1,15,6,3:wave 1,1,12,90,0:return
1030 for a=12 to 1 step-2:sound 1,15,a,5:wave 1,1,10,20
1040 next:return
1050 return
1060 wave 16,2,0,1000,3:return
1070 wave 16,2,0,18000,5:return
1080 for a=7 to 1 step-1:sound 1,15,1,a:wave 1,1,12,90,2:next
1090 sound 1,0,0,0:return
1100 rem char command
1110 gotoxy cx,cy:print m$:return
1120 gotoxy nx+q1,ly*4+ny+q2-(qt<15):print "o":return
```

©

The Works! For Commodore And Apple

James V. Trunzo

Requirements: Commodore 64 or 128 (in 64 mode) with a disk drive; or an Apple II-series computer with at least 64K RAM and a disk drive. Printer recommended.

"Jack of all trades but master of none" is a saying that could be applied to a collection of programs entitled *The Works!* from First Star Software. However, the saying would reflect only upon the level of sophistication of the individual programs making up *The Works!*, and should not be considered a criticism of the package as a whole.

The Works! is a compendium of useful programs for computer novices by Fernando Herrera, who first gained prominence with a program entitled *My First Alphabet*, winner of an Atari First Star Award. When considering *The Works!*, it is essential to keep in mind the audience for which it is intended.

The package is subtitled "A Complete Collection of Home Software," and that's just what *The Works!* is—with an emphasis on "home." It contains 13 programs divided into four main categories: Tools, Organizers, Arts, and Learning. Under the heading of Tools, you find such programs as Letter Writer, Loans & Investments, Calculator, Weights & Measures, and Math Formulas; the Organizers section includes Family Finances, Calendar Pad, Address Book, and Stock Portfolio; the Arts section has Graphics Painter and Music Composer; and the Learning section has Typing Teacher and Math Races.

All of the programs that make up *The Works!* are completely functional. However, none of them are—nor do they pretend to be—the final answer in their genre. Letter Writer, for example, is a more-than-adequate word processor for writing letters. It contains basic commands such as Move, Copy, Delete, and Insert, as well as a number of print-formatting commands. You wouldn't use it to do a college term paper with elaborate footnotes, though.

Similarly, all the other programs in *The Works!* provide a simple, working introduction to each type of software. They take advantage of the latest windowing techniques and are very easy to use. Also, there's a certain amount of integration among the programs. For example, when using Letter Writer, you can look up an entry in the Address Book program and merge it with a letter; or you can insert the result of a calculation done on the Calculator program into a report you are writing. You're also free to copy any of the programs onto a separate disk to be used apart from the rest, if you desire.

The value of *The Works!* lies in this: It gives new users a taste of a wide variety of programs and introduces some of the useful functions that can be performed by a home computer. Furthermore, it does so at a reasonable cost. You could pay $100 or more for a sophisticated financial program and then discover that you don't need it or don't like to use it. Family Finances in *The Works!* gives you the opportunity to try a program of this type before you invest in a more expensive package.

For those new to the world of computers and computer software, *The Works!* provides an easy entry into a sometimes bewildering domain.

The Works!
First Star Software, Inc.
18 East 41st Street
New York, NY 10017
$49.95

Under Fire For Apple

James V. Trunzo

Requirements: Apple II-series computer with at least 64K memory and a disk drive. A joystick is required for the Apple II+, but is optional on the Apple IIe and IIc. A version for the Commodore 64 and 128 is scheduled for release this spring.

If you're a war game buff and you've been waiting for the ultimate World War II infantry combat simulation, your wait may be over. Released by Avalon Hill, Ralph Bosson's *Under Fire* is an innovative milestone in all areas of computerized war gaming. It's one of the best war game simulations I've ever seen.

In fact, *Under Fire* is more than just a game: It's a complete, open-ended system in the same vein as its board game predecessor, the award-winning *Squad Leader*. With the three disks that come with *Under Fire*—a master disk, a scenario disk, and a mapmaker disk—you can design your own scenarios as well as play the standard games.

Although *Under Fire* is as complex as it is realistic, it is not a difficult game to learn (though playing and playing well are two different things). An extremely well-written rule book, complete with a step-by-step scenario, helps you get under way and allows you to absorb details bit by bit as you become more immersed in the mechanics of the game.

Under Fire lets you play solitaire or against another person, using one of nine prepared scenarios or one you have created yourself. You can assume command of men and weapons from the United States, Germany, or the Soviet Union. Each infantry squad, gun, and tank is individually represented on any one of three available maps: a situational map, showing the large-scale picture; a strategic map, depicting a smaller, more detailed portion of the battlefield; and a tactical map that shows the terrain and units to a degree of detail that is hard to believe. Frankly, an entire review could be written on this program's graphics alone.

Unprecedented Flexibility

Under Fire is so flexible that it truly lives up to Avalon Hill's boast that it's a "War Game Construction Set." When you combine the unit types, the terrain selections, the battlefield objectives, and the various orders of battle, there is almost no land engagement that cannot

be accurately simulated. That's why the nine scenarios provided are named after historical encounters; they represent types of conflicts ranging from open-field firefights to house-to-house battles. You can attack or defend objectives, recreate breakthroughs, or enter into all-out slugfests.

Lavish attention has been paid to details such as troop morale, training, supplies, skill levels, hidden units, line-of-sight fire, and animated combat. At the end of a battle, you get a complete report of men lost, men remaining, armor lost, and other statistics.

This review really just scratches the surface of *Under Fire*. For example, the Apple version even allows for the optional use of a Mockingboard to enhance the sound effects of raging battles. Avalon Hill and designer Bosson have created what is sure to become a standard for computerized war games in the future.

Under Fire
Avalon Hill Game Company
4517 Harford Road
Baltimore, MD 21214
$59.95

M-Disk For Atari ST

George Miller, Assistant Technical Editor

Requirements: Atari ST computer with at least 512K RAM; extra RAM recommended.

Do you often find yourself wishing for extremely fast, temporary storage to supplement the floppy disk drive on your Atari ST? *M-Disk*, by MichTron, lets you set aside a portion of Random Access Memory (RAM) as a RAM disk. This area of memory is used like a disk drive, except it's much faster. By moving small, frequently used programs or data files into the RAM disk, you can speed up access and make disk-intensive programs run more efficiently. (Of course, you still have to copy the programs or files you want to save onto a real disk before ending the session because the RAM disk disappears when power is shut off.)

M-Disk is easy to use. Michtron's user manual, although small, is well-written. Installing the RAM disk is no problem. And since the ST's operating system (TOS) sees the RAM disk as just another floppy disk drive, transferring files from memory or floppy disk to the RAM disk is a snap.

With *M-Disk*, you can specify the size of the RAM disk in 12K blocks up to a maximum of 800K, if your ST has this much memory available. Using the standard 520ST with 512K RAM, you can set up a RAM disk of 84K, or 111K if no GEM desktop accessories are loaded.

However, some larger application programs can't be used with a 512K machine and *M-Disk* due to memory limitations. For instance, it was almost impossible to use *M-Disk* with ST BASIC because BASIC normally leaves only about 5K free to begin with. If Atari carries through with plans to put TOS in Read Only Memory (ROM), more than 200K RAM will be freed up for such purposes.

When TOS is available in ROM, or when you add more memory, *M-Disk* will become an indispensible accessory for your Atari ST.

M-Disk
MichTron
576 S. Telegraph
Pontiac, MI 48053
$39.95

Atari XM301 Modem

Tom R. Halfhill, Editor

Requirements: Atari 400/800, XL, or XE computer with at least 48K RAM and a disk drive. The 1200XL requires a slight hardware modification (see text).

If you've been waiting for a (nearly) painless way to get started in telecomputing, the new Atari XM301 modem might be just the ticket. For about a quarter of what a bare-bones acoustic modem cost just a few years ago, the XM301 package includes a reliable, 300 bits-per-second (bps), direct-connect modem with autoanswer and autodial; an easy to use terminal program with full upload/download capabilities; free introductory time on popular commercial information services; and a well-written manual that guides you step by step through the often-confusing world of telecommunications.

Thanks to the latest modem-on-a-chip technology, the XM301 is just slightly larger and heavier than a pack of cigarettes. And that includes the power supply and interface, because the XM301 doesn't require a power supply and interface—it plugs directly into the Atari's serial input/output (SIO) port and draws its power from same. This is a major improvement over early Atari modems, which forced you to buy the $200 850 Interface Module and add yet another power transformer to the existing clutter.

Hooking up the XM301 takes just two steps. First, plug its permanently attached serial cable into the SIO port. Second, unplug the modular phone cord on your telephone and connect it to the XM301's modular jack.

The only complication is if you're using a 1200XL computer. The 1200XL designers wanted to discourage manufacturers from making peripherals that drew power from the computer, so they added a current-limiting resistor to the SIO port which keeps devices such as the XM301 from operating properly. Fortunately, the fix is not difficult for an Atari service technician or experienced electronic hobbyist. According to Atari, resistor R53 on the SIO port must be bypassed or replaced with a jumper. Atari recommends that you take your 1200XL into an Atari service center for this modification. (It won't affect any other operation of the computer.)

Upload And Download

Once the modem is hooked up, you're ready to run the terminal software included in the package, *XE Term*. Despite its name, *XE Term* works on 400/800 and XL series computers with at least 48K RAM as well as on the newer XE machines. The disk includes DOS 2.5 and an autoboot file that automatically runs *XE Term* when you switch on the computer.

Pop-up menus and single-keystroke commands make *XE Term* extremely easy to use. Yet it's not a stripped-down terminal emulator; it's actually a fairly versatile program that contains enough features to satisfy most people's telecomputing needs. Earlier Atari terminal programs, such as the *TeleLink I* and *TeleLink II* cartridges, often were criticized for their lack of file transfer functions. *XE Term* is a sharp departure from the *TeleLink* series. Not only does it allow you to upload and download files with other computers, it even provides three different protocols (file exchange schemes) for this purpose.

The simplest protocol is the ASCII transfer function. It's most often used for exchanging text files, such as documents created with a word processor. You can also "capture" any incoming text with this option and send it to the disk drive or printer. ASCII transfers don't employ any error-checking, however, so characters can get garbled if the phone line is noisy.

The second transfer protocol in *XE Term* is called XMODEM, a standardized scheme that is popular on many

152K *Lowest Price In The USA!* 152K

ATARI® Computer System Sale

• Students • Word Processing • Home • Business

LOOK AT ALL YOU GET FOR ONLY **$399** SYSTEM PRICE
LIMITED QUANTITIES

	LIST PRICE	INDIVIDUAL SALE PRICE
① Atari 130XE 152K Computer	$249.00	$134.95
② Atari 1050 127K Disk Drive	299.00	179.95
③ Atari 1027 Letter Quality 20 CPS Printer	299.00	179.95
Atari Writer Word Processer	59.95	49.95
Atari BASIC Tutorial Manual	16.95	12.95
TOTALS	**$923.90**	**$547.75**

All connecting cables & T.V. interface included.
☆ Monitors sold seperetly.

SAVE OVER $100
All 5 ONLY
$399.00
SYSTEM SALE PRICE

CALL FOR 1027 PRINTER REPLACEMENT OPTIONS

Other Accessories	List	Sale	
☆ 12" Hi Resolution Green Screen Monitor	$199.00	$79.95	Add $9.95 for Connection Cables
☆ 13" Hi Resolution Color Monitor	$399.00	$159.95	Add $10 for UPS

15 DAY FREE TRIAL. We give you 15 days to try out this ATARI COMPUTER SYSTEM!! If it doesn't meet your expectations, just send it back to us prepaid and we will refund your purchase price!! **90 DAY IMMEDIATE REPLACEMENT WARRANTY.** If any of the ATARI COMPUTER SYSTEM equipment or programs fail due to faulty workmanship or material within 90 days of purchase we will replace it IMMEDIATELY with no service charge!!

Best Prices • Over 1000 Programs and 500 Accessories Available • Best Service • One Day Express Mail • Programming Knowledge • Technical Support

Add $25.00 for shipping and handling!!
Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery. 2 to 7 days for phone orders 1 day express mail! We accept Visa and MasterCard. We ship C.O.D. to continental U.S. addresses only. Add $10 more if C.O.D., add $25 if Air Mail.

COMPUTER DIRECT

We Love Our Customers
22292 N. Pepper Rd., Barrington, Ill. 60010
312/382-5050 to order

Famous Smith Corona National Brand

10" PRINTER SALE

Below Wholesale Cost Prices!!!

• ONE YEAR IMMEDIATE REPLACEMENT WARRANTY

- **Speed: 120 or 160 characters per second**
- **Friction Feed/Tractor Feed — Standard**
- **80 character print line at 10 CPI**
- **1 Line Buffer, 2K Buffer on 120/160 CPS Plus LQM**
- **Six pitches**
- **Graphics capability**
- **Centronics compatible parallel interface**
- **Features Bidirectional Print, Shortline Seek, Vertical And Horizontal Tabs**

SUPER GRAPHICS

This is a sample of our emphasized near-letter-quality print.

italic print. There is standard data processing quality print

Check these features & prices

120 CPS 10" Printer

List $429.00 **SALE $159**

120 CPS + Letter Quality Mode 10" Printer

List $449.00 **SALE $179**

160 CPS + Letter Quality Mode 10" Printer

List $499.00 **SALE $199**

SPECIFICATIONS

(IBM — Commodore)

Size/Weight
Height 5.04" Width 16.7"
Depth 13.4" Weight 18.7 lbs.
Internal Char. Coding
ASCII Plus ISO
Print Buffer Size
120 CPS: 132 Bytes (1 line)
120/160 CPS Plus LQM: 2K
No. of Char. In Char. Set
96 ASCII Plus International
Graphics Capability
Standard 60, 72, 120 DPI
Horizontal 72 DPI Vertical
Pitch
10, 12, 16.7, S, 6, 8.3, Proportional Spacing
Printing Method
Impact Dot Matrix

Char. Matrix Size
9H x 9V (Standard) to 10H x 9V
(Emphasized & Elongate)
Printing Features
Bi-directional, Short line seeking, Vertical Tabs, Horizontal Tabs
Forms Type
Fanfold, Cut Sheet, Roll (optional)
Max Paper Width
11"
Feeding Method
Friction Feed Std., Tractor Feed Std.
Ribbon
Cassette — Fabric inked ribbon
Ribbon Life
4 million characters

(Apple — Atari — Etc.)

Interfaces
Parallel 8 bit Centronics compatible
120/160 CPS Plus NLQ: RS232 Serial inc.
Character Mode
10 x 8 Emphasized, 9 x 8 Standard; 10 x 8 Elongated; 9 x 8 Super/Sub Script (1 pass)
Character Set
96 ASCII
11 x 7 International Char.
Line Spacing
6/8/12/72/144 LPI
Character Spacing
10 cpi normal, S cpi elongated normal; 12 cpi compressed, 6 cpi elongated compressed; 16.7 cpi condensed; 8.3 cpi elongated condensed; S.12.S cpi elongated proportional

Cartridge Ribbon — List $19.95. **Sale $12.95.**

Interfaces

IBM $89.00 **Apple $59.00** **Atari $59.00** **Commodore $39.00**

Add $14.50 for shipping, handling and insurance. Illinois residents please add 6% tax. Add $29.00 for CANADA, PUERTO RICO, HAWAII, ALASKA, APO-FPO orders. Canadian orders must be in U.S. dollars.
WE DO NOT EXPORT TO OTHER COUNTRIES, EXCEPT CANADA.
Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days delivery. 2 to 7 days for phone orders. 1 day express mail!
VISA — MASTERCARD — C.O.D. No C.O.D. to Canada or APO-FPO

COMPUTER DIRECT
We Love Our Customers
22292 N. Pepper Rd., Barrington, Ill. 60010
312/382-5050 to order

COMMODORE 64 COMPUTER

(Order Now)

$139.95

- C128 Disks 79¢ ea.*
- Paperback Writer 64 $34.95
- 10" Comstar 10X Printer $148.00
- 13" Zenith Color Monitor $139.95

CALL BEFORE YOU ORDER

COMMODORE 64 SYSTEM SALE

Commodore 64
Com. 1541 Disk Drive
13" Color Monitor

Plus $30.00 S&H

$457

PLUS FREE $49.95 Oli Barons Adventure Program

C128 COMMODORE COMPUTER

NEW

(Order Now)

$289.00

Plus FREE $69.95 Timeworks Wordprocessor.

- 340K 1571 Disk Drive $259.00
- Voice Synthesizer $39.95
- 12" Amber Monitor $79.95

PRICES MAY BE LOWER

SPECIAL SOFTWARE COUPON

We pack a SPECIAL SOFTWARE DISCOUNT COUPON with every COMMODORE 64 COMPUTER, DISK DRIVE, PRINTER, or MONITOR we sell! This coupon allows you to SAVE OVER $250 OFF SALE PRICES!!

(Examples)

PROFESSIONAL SOFTWARE COMMODORE 64

Name	List	Sale	Coupon
Paperback Writer 64	$99.00	$39.95	$29.95
Paperback Database 64	$69.00	$34.95	$24.95
Paperback Dictionary	$24.95	$14.95	$10.00
The Print Shop	$44.95	$27.95	$26.95
Halley's Project	$39.95	$25.95	$24.95
Practicalc (spread sheet)	$59.95	$19.95	$14.95
Programmers Reference Guide	$24.95	$16.95	$12.50
Nine Princes in Amber	$32.95	$24.95	$21.95
Super Bowl Sunday	$30.00	$19.95	$17.95
Flip & File Disk Filer	$24.95	$14.95	$12.95
Deluxe Tape Cassete (plus FREE game)	$89.00	$44.95	$34.95
Fra Joystick	$19.95	$12.95	$10.00
Computer Care Kit	$44.95	$29.95	$24.95
Dust Cover	$ 8.95	$ 6.95	$ 4.60
Injured Engine	$39.95	$27.95	$24.95
Pitstop II (Epyx)	$39.95	$22.95	$19.95
Music Calc	$59.95	$14.95	$12.95
File Writer (by Codewriter)	$39.95	$29.95	$24.95

(See over 100 coupon items in our cotolog)

Write or call for Sample SPECIAL SOFTWARE COUPON!

COMMODORE 64 COMPUTER $139.95

You pay only $139.95 when you order the powerful 84K COMMODORE 64 COMPUTER! LE55 the value of the SPECIAL SOFTWARE DISCOUNT COUPON we pack with your computer that allows you to SAVE OVER $250 off software sale prices!! With only $100 of savings applied, your net computer cost is $39.95!!

* C128 DOUBLE SIDED DISKS 79¢ EA.

Get these 5¼" Double Sided Floppy Disks specially designed for the Commodore 128 Computer (1571 Disk Drive) 100% Certified, ***Lifetime Warranty***, Automatic Lint Cleaning Liner included. 1 Box of 10 - $9.90 (99¢ ea.), 5 Boxes of 10 - $44.50 (89¢ ea.), 10 Boxes of 10 - $79.00 (79¢ ea.)

13" ZENITH COLOR MONITOR $139.95

You pay only $139.95 when you order this 13" ZENITH COLOR MONITOR. LE55 the value of the SPECIAL SOFTWARE DISCOUNT COUPON we pack with your monitor that allows you to save over $250 off software sale prices!! With only $100 of savings applied, your net color monitor cost is only $39.95. (16 Colors)

Premium Quality 120-140 CPS Comstar 10X Printer $148.00

The COMSTAR 10X gives you a 10" carriage. 120-140 CPS, 9 x 9 dot matrix with double strike capability for 18 x 18 dot matrix (near letter quality), high resolution bit image (120 x 144 dot matrix), underlining, back spacing, left and right margin setting, true lower decenders with super and subscripts, prints standard, italic, block graphics and special characters. It gives you print quality and features found on printers costing twice as much!! (Centronics Parallel Interface) List $399.00 **Sale $148.00.**

4 SLOT EXPANDER & 80 COLUMN BOARD $59.95

Now you program 80 COLUMNS on the screen at one time! Converts your Commodore 64 to 80 COLUMNS when you plug in the 80 COLUMN EXPANSION BOARD!! PLUS 4 slot expander! ***Limited Quantities***

80 COLUMNS IN COLOR PAPERBACK WRITER 64 WORD PROCESSOR $39.95

This PAPERBACK WRITER 64 WORD PROCESSOR is the finest available for the COMMODORE 64 computer! The ULTIMATE FOR PROFESSIONAL Word Processing, DISPLAYS 40 or 80 COLUMNS IN COLOR or black and white! Simple to operate, powerful text editing, complete cursor and insert/delete key controls line and paragraph insertion, automatic deletion, centering, margin settings and output to all printers! List $99.00 **SALE $39.95** Coupon $29.95

ATTENTION

Computer Clubs

We Offer Big Volume Discounts

CALL TODAY!

PROTECTO WARRANTY

All Protecto's products carry a minimum 90 day warranty. If anything fails within 90 days from the date of purchase simply send your product to us via United Parcel Service prepaid. We will IMMEDIATELY send you a replacement at no charge via United Parcel Service prepaid. This warranty proves once again that ***We Love Our Customers.***

C128 COMMODORE COMPUTER $289.00

We expect a limited supply for Christmas. We will ship on a first order basis. This all-new revolutionary 128K computer uses all Commodore 64 software and accessories plus all CPM programs formatted for the disk drive. ***Plus FREE $69.95 Timeworks Wordprocessor.***
List $349.00 **SALE $289.00.**

340K 1571 COMMODORE DISK DRIVE $259.00

Double Sided, Single Disk Drive for C-128 allows you to use C-128 mode plus CPM mode. 17 times faster than 1541, plus runs all 1541 formats.
List $349.00 **Sale $259.00.**

SUPER AUTO DIAL MODEM $29.95

Easy to use. Just plug into your Commodore 64 computer and you're ready to transmit and receive messages. Easier to use than dialing your telephone, just push one key on your computer! Includes exclusive easy to use program for up and down loading to printer and disk drives. ***Best In U.S.A.***
List $99.00 **SALE $29.95.** *Coupon $24.95*

VOICE SYNTHESIZER $39.95

For Commodore-64 computers. Just plug it in and you can program words and sentences, adjust volume and pitch, make talking adventure games, sound action games and customized talkies!! PLUS ($19.95 value) TEXT TO SPEECH program included FREE, just type a word and hear your computer talk — ADD SOUND TO "ZORK", SCOTT ADAMS AND OTHER ADVENTURE GAMES!! (Disk or tape.) List $89.00 **SALE $39.95**

12" MAGNAVOX (NAP) 80 COLUMN MONITOR WITH SOUND $79.95

Super High Resolution green screen monitor. 80 columns x 24 lines, easy to read, plus speaker for audio sound included. Fantastic value. List $129.00 **Sale $79.95.** (C128 cable $19.95, C64, Atari cable $9.95)

PRINTER/TYPEWRITER COMBINATION $229.95

"JUKI" Superb letter quality, daisy wheel printer/typewriter combination. Two machines in one — just a flick of the switch. 12" extra large carriage, typewriter keyboard, automatic margin control and relocate key, drop in cassette ribbon! (90 day warranty) centronics parallel or RS232 serial port built in (Specify). List $349.00, **SALE $229.95.** **(Ltd. Qty.)**

13" RGB & COMPOSITE COLOR MONITOR $259.95

Must be used to get 80 columns in color with 80 column computers (C128 - IBM - Apple).
(Add $14.50 shipping)
List $399.00 **SALE $259.95.**

- LOWEST PRICES • 15 DAY FREE TRIAL
- BEST SERVICE IN U.S.A. • ONE DAY EXPRESS MAIL

PHONE ORDERS
8 a.m. - 8 p.m. Weekdays
9 a.m. - 12 noon Saturdays

- 90 DAY FREE REPLACEMENT WARRANTY
- OVER 500 PROGRAMS • FREE CATALOGS

Add $10.00 for shipping, handling and insurance. Illinois residents please add 6% tax. Add $20.00 for CANADA, PUERTO RICO, HAWAII, ALASKA, APO-FPO orders. Canadian orders must be in U.S. dollars. WE DO NOT EXPORT TO OTHER COUNTRIES EXCEPT CANADA. Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery. 2 to 7 days for phone orders. 1 day express mail! VISA — MASTER CARD — C.O.D. No C.O.D. to Canada, APO-FPO

PROTECTO

We Love Our Customers

Box 550, Borrington, Illinois 60010

312/382-5244 to order

electronic bulletin board systems (BBSs). XMODEM is somewhat slower than ASCII protocol, but it checks for transmission errors during the transfer. This makes it particularly useful for uploading and downloading such critical files as programs.

XE Term's third transfer scheme was a surprise—the A-protocol used by the CompuServe Information Service. *XE Term* automatically recognizes A-protocol, so it's easy to download programs and other files from CompuServe.

The only drawback to *XE Term*'s file transfer capabilities is its rather small buffer—only about 14K. Most Atari terminal programs have larger buffers. This restriction means that files longer than 14K must be broken up into pieces before uploading or downloading.

Online Phone Book

XE Term strikes a balance between ease of use and full-featured telecommunications. You can link up with Ataris, most other personal computers, information services, and BBSs of almost any flavor. But if the other computer is an oddball or a mainframe with unusual requirements, you may need a program that provides more communication options.

To reach *XE Term*'s Options menu, you press O from the main Functions menu (a keystroke that Atari forgot to list on the Functions menu, by the way). The Options menu pops up and lets you change the input parity (none, even, odd, or clear); output parity (none, even, odd, or set); the duplex mode (half or full); and translation mode (ASCII and Atari ASCII, called ATASCII). You can also change the left screen margin from its normal indentation of two characters.

A few convenience features make *XE Term* even easier to use. You can save the communication settings to disk in a configuration file so the program always boots up the way you want it to. A dialing menu lets you switch between tone and pulse dialing, store up to five phone numbers that can be dialed with a single keystroke, or dial directly from the computer keyboard. When you're establishing a connection, all telephone sounds are routed through the TV or monitor speaker. That way, you can listen to the modem dialing the number and hear any busy signals, recorded messages, or humans that may be encountered on the other end of the line.

Automatic log-on sequences can be programmed to speed up access to remote computer systems. For instance, you can tell *XE Term* to dial your local CompuServe access number, type a CTRL-C to get CompuServe's attention, and enter your ID number and password—all without any intervention on your part. You can also set up the modem for autoanswer mode in case someone wants to call your computer.

A Few Extros

The 50-page manual is very well-written and contains just about everything you need to know to use the XM301 modem and *XE Term*. There's even a short glossary of telecommunication terms. For advanced users, a file on the *XE Term* disk contains technical information about the modem and its software interface. Another disk file, HANDLER.OBJ, is the modem's device driver. This makes it possible for programmers to write their own custom terminal or BBS software for the XM301.

On top of all this, you get introductory offers to online services and discount coupons that by themselves are worth more than the cost of the XM301 package. There's $15 worth of free time on CompuServe, a $20 discount for joining The Source, a $75 password and hour's time on Dow Jones News/

Amplify your skills

WITH THESE NEW INTRODUCTORY BOOKS FROM COMPUTE! BOOKS.

These titles will help you unleash the power inside your computer. Whether you're an experienced programmer learning a new language or a beginner just starting out, these books will show you, clearly and quickly, how to get more than you ever thought possible from your computer.

THE AMAZING AMIGA

The Amiga: Your First Computer
Dan McNeill
Written in a lively and entertaining style, this book teaches everything a beginner needs to know to get started quickly with the Amiga from Commodore. You'll learn about setting up the system, some of the most popular types of software, and details about the hardware.
ISBN 0-87455-025-4
$16.95

Using AmigaDOS
Arlan R. Levitan and Sheldon Leemon
A comprehensive reference guide and tutorial to the powerful AmigaDOS—the operating system underlying the Workbench and Intuition—this book offers information useful to every Amiga owner. AmigaDOS, the alternative to the icon-based Workbench, lets you control the computer directly. Using AmigaDOS defines and illustrates all DOS commands, and shows you how to create file directories, access peripherals, and run batch file programs. You'll learn why the system prompts you to swap disks, and how to avoid "disk shuffle." The screen- and line-oriented text editors, both overlooked in the user's guide which comes with the Amiga, are explained in detail. Numerous examples and techniques show you how to use AmigaDDS to make operating your computer even more convenient. A full reference section details each DOS command, giving you easy access to the complete AmigaDDS.
ISBN 0-87455-047-5
$14.95

BRING THE ATARI ST ALIVE

Introduction to Sound and Graphics on the Atari ST
Tim Knight
The ST, Atari's powerful new computer, is an extraordinarily impressive sound and graphics machine. Easy to use, the ST can produce color graphics and sound. You'll find thorough descriptions of the computer's abilities, and the information needed to create a complete sound and graphics system. This is the perfect introductory reference to sound and graphics on the Atari ST.
ISBN 0-87455-035-1
$14.95

LEARN C

From BASIC to C
Harley M. Templeton
This introduction to C takes you by the hand and shows how to move from BASIC to this increasingly popular language. BASIC programmers will find this approach designed just for them. Early chapters discuss C language equivalents for common BASIC statements and the similarities and differences between BASIC and C. Later chapters teach everything you need to know to write, debug, and compile programs in C.
ISBN 0-87455-026-2
$16.95

Visit your local bookstore or computer store to purchase any of these new, exciting books from COMPUTE! Publications. Or order directly from COMPUTE!. Call toll free 1-800-346-6767 (in NY 212-265-8360) or mail your check or money order (including $2.00 shipping and handling per book) to COMPUTE! Books, P.O. Box 5038, F.D.R. Station, New York, NY 10150.

COMPUTE! Publications, Inc.
One of the ABC Publishing Companies
825 7th Avenue, 6th Floor, New York, NY 10019
Publishers of COMPUTE! COMPUTE!'s Gazette, COMPUTE!'s Gazette Disk, COMPUTE! Books, and COMPUTE!'s Apple Applications

COMPUTE! books are available in the U.K., Europe, the Middle East, and Africa from Holt Saunders, Ltd., 1 St. Anne's Road, Eastbourne, East Sussex BN21 3UN, England and in Canada from Holt, Rinehart, & Winston, 55 Horner Ave., Toronto, ON M8Z 4X6.

Retrieval, a $10 coupon for joining Dialog's Knowledge Index, and a free password and month's access to Dun & Bradstreet's electronic edition of the Official Airline Guides.

The Atari XM301 package is definitely one of today's best values in telecomputing.

Atari XM301
Atari Corporation
1196 Borregas Avenue
Sunnyvale, CA 94088
$49.95

EduCalc And NoteCard Maker

Karen G. McCullough

Requirements: Commodore 64 or 128 (in 64 mode) with a disk drive; Apple II-series computer with at least 64K RAM and a disk drive; or an IBM PC/PCjr with a disk drive. The Apple version was reviewed.

Recent attempts to teach computer literacy have focused on familiarizing students with the software tools commonly used on personal computers. To assist that effort, Grolier Electronic Publishing, Inc. has released two programs: *EduCalc* and *NoteCard Maker*. Each is a simplified version of a popular type of software for personal computers and is aimed at junior to senior high school students.

EduCalc is a spreadsheet—a stripped-down, bare-bones version of programs like *VisiCalc*, *Multiplan*, and *Lotus 1-2-3*. Spreadsheets allow computers to perform sophisticated mathematical operations on large quantities of numbers. As with all spreadsheets, *EduCalc* lets you enter numbers, mathematical formulas, or text in rows and columns on the screen. Each piece of information is stored in a cell identified by a letter for the column and a number for the row. The numbers entered into the cells can be manipulated in various ways by other cells containing formulas. The result of a particular operation is displayed in that formula's cell.

EduCalc includes a tutorial which serves as an excellent introduction to spreadsheets in general as well as to *EduCalc*. It's simple, clear, and should make novices comfortable with the program in 15 to 20 minutes. A practice template lets you experiment with the program. Unlike more powerful spreadsheets intended for professional business applications, *EduCalc* is extremely easy to use. Menus guide you through lists of options, and various functions are displayed on the screen along with advice on using them. The program does a good job of protecting you against errors.

No Shortcuts

But there's a price for that simplicity. Where *EduCalc* is friendly to novice users, it may frustrate those who become more experienced with it. The program structure is rigid—there are no shortcuts.

For example, to enter a formula to add up a column of numbers, you choose Enter from the Tool menu, move the highlighted cursor to the cell where the results are to be displayed, choose Formula from the Entry menu, and press S for sum. Then you move the cursor to the first cell of the column to be totaled, press RETURN, move the cursor to the last cell of the column, and press RETURN again. It sounds easy and logical, but it's also frustratingly time-consuming if the column is 40 figures long.

There are other limitations as well. Only mathematical operations can be performed—there are no logical or lookup functions. And you can't jump directly to a specific cell on the spreadsheet—you must move there with the cursor. That can be very slow on a large sheet, because the program redraws the screen each time the cursor moves off the edge.

The *EduCalc* manual could use larger print, an index, and better explanations. For example, when you're saving a spreadsheet on disk for the first time, the manual says the program should ask if you want to initialize the disk. But the program doesn't. Fortunately my disk was already initialized, so I had no problem reloading the spreadsheet.

A Quick Organizer

NoteCard Maker is a simplified database manager program. As its name implies, *NoteCard Maker* is intended to help students collect and organize information, especially when writing reports or term papers. It transforms the screen into a series of electronic note and bibliography cards. After entering information onto the cards, students can search for specific items or sort them in various ways.

The tutorial that comes with *NoteCard Maker* is just as effective as *EduCalc*'s. Most junior high school students will be comfortable with the program after 20 minutes' work. And if they forget what to do at any point while entering information, they can simply press CTRL-A to bring up a screenful of advice.

The process of entering information and editing the cards is simple and straightforward, and once the cards are created, there are plenty of options for using them. Both notecards and bibliography cards can be searched, sorted, viewed on screen, and listed on a printer.

Like *EduCalc*, *NoteCard Maker*'s main drawback is rigid structure. There are only three options for file size, and the size can't be changed once the file is created. Nor can you alter the format of the cards or bibliographic information. Also, *NoteCard Maker* lets you create a duplicate file without warning that a file of the same name already exists.

Both *EduCalc* and *NoteCard Maker* are excellent programs for introducing students or novices to spreadsheets and database managers. They also may be the solution if you need a simple spreadsheet or database without a lot of complex commands. For these purposes, both programs are effective tools.

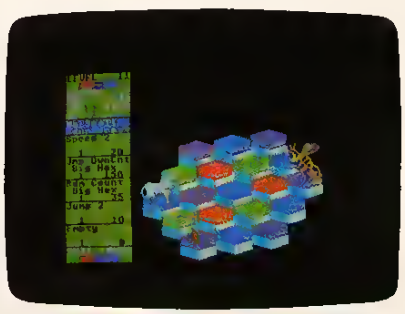
EduCalc
NoteCard Maker
Grolier Electronic Publishing, Inc.
95 Madison Avenue
New York, NY 10016
$49.95 (EduCalc)
$59.95 (NoteCard Maker)

Hex For Atari ST

George Miller,
Assistant Technical Editor

Requirements: Atari ST computer and a joystick.

Colors swirl about you, constantly changing patterns as the arena is affected first by your magic, then by the magical powers of your opponent. Might this be the day that you divine the secrets of this mystical realm and emerge victorious, hailed by all as the most powerful magician in the universe?



ATARI 130XE

ATARI 130XE Super Computer Package
130XE Computer
1050 Disk Drive
1027 Printer
Atariwriter
CALL

ATARI PRINTER INTERFACES

Uprint A	54.95
Uprint AW/16K	79.95
Uprint AW/64K	99.95
MPP 1150	59.95

INDUS GT DISK DRIVE... Call

ATARI 130XE SUPER PRINTER PKGS.

SG-10 Printer and U-Prnt A	279
Panasonic 1091 and U-Print A	309

Super Printer Packages have no extra shipping charges or credit card surcharges when shipped in Continental USA

ATARI 130XE SOFTWARE

BRODERBUND

Print Shop	28.95
Karateka	20.95
Print Shop Graph I, II or III	19.95
Print Shop Comp	27.95

INFOCOM
See Commodore 64 section for items and prices

ELECTRONIC ARTS

Archon II	24.95
Archon	19.95
Seven Cit of Gold	24.95
Skyfox	24.95
Pinball Const	24.95
One on One	24.95

MICROPROSE

Silent Service	23.95
Gunship	23.95
Acrojet	23.95
F-15 Strike Eagle	23.95
Kennedy Approach	23.95

OSS

Basic XE Cart	52.95
MAC 65 XL-Cart	49.95
Action-Cart	49.95
Basic XL Cart	39.95
All Tool Kits	20.95

BATTERIES INCLUDED

Home Pak	34.95
Paper Clip	39.95
B-Graph	34.95

SYNAPSE

Syncalc	32.95
Synfile	32.95
Syntrend	25.95
Syncalc Templates	16.95
Loderunner Rescue	20.95
Mindwheel	27.95
Essex	27.95
Brimstone	27.95

SSI
See Commodore 64 section for items and prices

MISCELLANEOUS 130XE

Hacker	19.95
Amer Cross Ctry	19.95
Flight Simulator II	34.95
Ultima II	37.95
Ultima III	37.95
Universe	69.95
Letter Perfect	39.95
Data Perfect	39.95
Halley Project	27.95
Ultima I	23.95
Ultima IV	41.95
MMG Basic Comp	69.95

ATARI 520ST*

Atari 520ST-RGB System...Call
Atari 520ST-Monochrome Sys...Call
SF314DS/DD 1 Megabyte Disk Drive.....Call
We warranty all 520ST computers purchased from ComputAbility for ninety days
***Please call for stock availability on Atari ST products before ordering by mail.**

ATARI 520ST SOFTWARE

HABA SYSTEMS

Hippo C	54.95
Haba Write	54.95
Haba Calc	54.95
Haba Graph	54.95
Haba File	54.95

MISCELLANEOUS ST

ViP Professional	129.95
Ultima II	39.95
Perry Mason	34.95
Degas	27.95
Farenheit 451	34.95
Amazon	34.95
Hacker	29.95
The Final Word	94.95
Deja Vu	39.95
PC/Intercom	69.95
Hex	27.95
Crimson Crown	27.95
Mudpies	23.95
King's Quest II	Call
Gato	Call
Borrowed Time	34.95
Personal Prolog	79.95
Personal Pascal	64.95
Zoomracks	Call
Mi Term	54.95
Regent Word	34.95

INFOCOM ST

Deadline	34.95
Starcross	34.95
Zork I, II or III	29.95
Witness	27.95
Suspended	34.95
Planetfall	27.95
Sorcerer	29.95
Seastalker	27.95
Cutthroats	27.95
Hitchiker	27.95
Suspect	29.95
Wishbringer	27.95
Infidel	29.95
Enchanter	27.95
Spellbreaker	34.95
Mind Forever Voy	34.95

AMIGA

AMIGA PERSONAL computer*
Call for Herdware and add-on peripherale prices

AMIGA SOFTWARE

Archon	27.95
Artic Fox	27.95
Deluxe Paint	59.95
Fin Cookbook	34.95
One on One	27.95
Seven Cities	27.95
Skytox	27.95
Hacker	29.95
Mindshadow	29.95

***Please call for availability of Amiga products before ordering by mail.**

EST 1982

ComputAbility

We stock hundreds of programs for the Apple, Atari, C-64 and IBM.
If you don't see it listed here, don't hesitate to call.

No Surcharge for MasterCard/Visa.

VISA MasterCard

APPLE PRINTER INTERFACES AND BOARDS

Apricorn Parallel w/Graphics	69.95
Apricorn 16K Expansion Board	82.95
Apricorn 80 Column Brd	64.95
Apricorn RS232 Interface	69.95
U-Print-Apple IIC w/64K	109.95
U-Print-Apple IIC w/16K	89.95

LASER 3000 PERSONAL COMPUTER

Apple 2+ Compatible
Built-in Microsoft Basic in ROM
Built-in RGB and Composite Video
Built-in 80 Column Display
Built-in Centronics Printer Interface
Built-in Numeric Keypad
192K RAM
Bundled Productivity Software
Disk Drive
All this and more for the amazing **LOW** price

$399

APPLE SOFTWARE

BRODERBUND

Print Shop	33.95
Print Shop Graphics I, II or III	17.95
Print Shop Comp	27.95
Karateka	21.95
Carmen Sandiego	25.95
Science Toot Kit	39.95
Bank Street Writer	44.95
Fantavision	Call

ELECTRONIC ARTS

Adventure Const	34.95
Archon II	27.95
Bard's Tale	29.95
Imagic Football	24.95
Auto-Duel	34.95
Skytox	27.95
Lords of Conquest	27.95
One on One	27.95
Ultima III	39.95
Ultima IV	39.95

EPYX

Battlazer	24.95
Winter Games	24.95
Summer Games II	24.95
World's Great/ Football	24.95

INFOCOM
See Atari 520ST section for items and prices.

MICROPROSE
See Atari 130XE section for items and prices

MINDSCAPE

Color Me	20.95
Crossword Magic	34.95
Halley Project	29.95
A View To Kill	27.95
Racter	29.95
The Mist	27.95
Perfect Score	49.95
Voodoo Island	27.95

DAVIDSON

Math Blaster	34.95
Alge-Blaster	34.95
Spell It	34.95
Word Attack	34.95

SIMON & SCHUSTER

Typing Tutor III	34.95
Kobiashi Adv	29.95
Webster Spell Chk	34.95
Webster Thesaurus	84.95
Lovejoy SAT	49.95

SIR-TECH

Wizardry/Oiam	23.95
Wizardry/Legacy	27.95
Wizardry/Proving	33.95
Wizardry/Wernda	29.95
Wizprint	19.95

SSI
See Commodore 64 section for items & prices.

APPLE MISCELLANEOUS

Beach-Head	23.95
Beach-Head II	23.95
Gamemaker	27.95
Hacker	27.95
Hardball	24.95
Sundog	27.95
Paper Clip	64.95
OLM Software	Call
The Works	34.95
Star League Base	23.95
Educalc	34.95
Friendly Filer	27.95
Note Card Maker	27.95
Hayden Software	Call
Microleague Base	29.95
Random House	Call
PFS Software	Call
Newsroom	39.95
Gato	27.95
Weekly Reader	Call

COMMODORE 128

C-128 Computer	Call
1571 Disk Drive	Call
1902 Monitor	Call
1670 Modem	Call

Westridge 6470 300/1200 Modem 179.95

IBM PC

IBM PC SYSTEMS
Configured to your specific needs
Call for lowest price on IBM-PC, IBM-XT or IBM-AT

Corona PC-400 Compatible Call
Corona Portable PC Compatible Call

PC Multifunction Boards
We carry the complete line of **AST, Hercules, Paradise, STB, and Quadram Call for current prices**

IBM PC SOFTWARE

Print Shop	39.95
Print Shop Graph I	27.95
Bank Street Writer	49.95
Ancient Art of War	29.95

BORLAND

Sidekick	37.95
Turbo Pascal	49.95

BLUE CHIP

Baron	34.95
Squire	34.95
Millionaire	34.95
Tycoon	34.95

DIGITAL RESEARCH
Call for items and prices

INFOCOM
See Atari 520ST for items and prices

LEADING EDGE

Nutshell	69.95
LE/WP Basic	67.95
LE/Word Proc +Speller	169.95

MICROPROSE

F-15 Strike Eagle	23.95
Kennedy Approach	27.95
Acrojet	27.95
Silent Service	27.95

MICROSOFT

Flight Simulator	38.95
Word	249.00
Multiplan	134.95

MINDSCAPE
See Apple Section for items and prices

SIERRA

King's Quest	34.95
King's Quest II	34.95
Ultima II	39.95

THOUGHTWARE
Call for items and prices

IBM MISCELLANEOUS

PFS	Call
Gato	27.95
Wizardry	39.95
Strip Poker	27.95
Electric Desk	204.95
D-Base III	Call
Sideways	39.95
Home Pak	34.95
Sargon III	34.95
Peachtree	Call
Jet	34.95
BPI Business	Call
Newsroom	39.95

COMMODORE 128 SOFTWARE

Multiplan 64/128	44.95
Consultant	52.95
Paper Clip +Spell	54.95
Swiftcalc	49.95
Wordwriter	49.95
Data Manager	49.95
Fleet System II	44.95
Superbase 128	69.95
Mach V/128	34.95

C-64 SUPER COMPUTER PACKAGE

C-64 Computer
1541 Disk Drive
803 Printer
ONLY $399

C-64 SUPER PRINTER PKGS.

SG-10 & G-Wiz	279
Panasonic 1091 & G-Wiz	309
Legend 1080 & G-Wiz	269

If Xetec Supergraphic is desired add $10 to package.
Super Printer packages have no added shipping or charge card surcharges added when shipped in Continental USA

Wesfridge AA/AD Modem . . 49.95
Cardco G-Wiz . . . 54.95

GENERAL HARDWARE

Star Micronics Inc. The Power Behind The Printer Work

SG-10	215
SG-15	369
SD-10	339
SD-15	449
SR-10	489
SR-15	Call

PRINTERS

Panasonic 1091	245
Legend 808	169
Legend 1080	209
Powertype	309
Juki 5510	389
Epson	Call

PRINTER BUFFERS

Microfazer	From 169
U Butt 16K	79.95
U Butt 64K	99.95

MODEMS

US Robotics 2400	469
Volksmodem 1200	189
Prometheus 1200	319
Password 1200	209
Novation	Call
PC Modems	Call

MONITORS

Technika MJ 22RGB	269
Sakata SC-100	169
Samsung 12" Green	79.95
Samsung 12" Amb	79.95
Taxan	Call
Amdek	Call
Commodore 1802	169

COMMODORE 64 SOFTWARE

ACCESS

Beach-Head	21.95
Beach-Head II	24.95
Raid/Moscow	24.95
Mach V-Cart	21.95

INFOCOM

Zork I	24.95
Zork II or III	27.95
Deadline	29.95
Starcross	29.95
Witness	29.95
Planetfall	24.95
Hitchiker	24.95
Enchanter	24.95
Cutthroats	24.95
Sorcerer	29.95
Spellbreaker	29.95

SSI

Battallion Commander	24.95
Battle of Antietnam	32.95
Fighter Command (No Atari)	37.95
Norway 85 (No Atari)	21.95
Panzer Grenidier	24.95
USAAF	37.95
Breakthrough/ Ardennes	37.95
Kampgruppe	37.95
Phantasie (No Atari)	24.95
Broadsides	24.95
Carrier Force	37.95
Comp Ambush	37.95
Mech Brigade (No Atari)	37.95
Field of Fire (No Apple)	24.95
Op Mkt Garden	32.95
Pro Tour Golf (No Atari)	24.95
Gemstone Warrior	21.95
Imp Galactum	24.95
Computer Baseball	24.95
Comp Quarterback	24.95

ELECTRONIC ARTS

Adv Construction	29.95
Mail Order Monster	24.95
Racing Destruction	24.95
Ultima III	37.95
Ultima IV	41.95
Bard's Tale	27.95

See Atari 130XE section for rest of items & prices

EPYX

Eidolon	24.95
Koronis Rift	24.95
Winter Games	24.95
Apshai Trilogy	24.95
Fast Load Cart	24.95
Program/Tool Kit	29.95

MICROPROSE
See Atari 130XE section for items and prices

MISCELLANEOUS COMMODORE 64

Print Shop	28.95
Cal-Kit	34.95
Superbase 64	47.95
Karateka	20.95
Hacker	20.95
Gamemaker	27.95
Ultima II	37.95
Karate Champ	25.95
Essex	27.95
Kung Fu/ Kung Fu Master	25.95
9 Princes of Amber	22.95
Spy vs Spy Vol II	23.95
Maxi Golf	24.95
Internat'l Hockey	19.95
The Fourth Proto	23.95
Blazing Paddles	24.95
Mirage Word	34.95
Mirage Database	34.95
Welcome Aboard	19.95
Super Huey	14.95
Spell It	34.95
Math Blaster	34.95
Word Attack	34.95
Odesta Chess	49.95
Brimstone	27.95

EST. 1982

ComputAbility™

PO Box 17882, Milwaukee, WI 53217
ORDER LINES OPEN
Mon-Fri 11 a.m. - 7 p.m. CST • Sat. 12 p.m. - 5 p.m. CST

To Order Call Toll Free
800-558-0003

For Technical Info., Order Inquiries, or for Wisc. Orders
414-351-2007

ORDERING INFORMATION: Please **specify system**. For fast delivery send cashier's check or money order. Personal and company checks allow 14 business days to clear. School P.O.'s welcome. **C.O.D. charges are $3.00**. In Continental U.S.A. include $3.00 for software orders. 4% shipping for hardware, minimum $4.00. MasterCard and Visa orders please include card #, expiration date and signature. WI residents please include 5% sales tax. HI, AK, FPO, APO, Puerto Rico and Canadian orders please add 5% shipping, **minimum $5.00**. All other **foreign orders add 15% shipping, minimum $10.00**. All orders shipped outside the Continental U.S.A. are shipped first class insured U.S. mail. If foreign shipping charges exceed the minimum amount, you will be charged the additional amount to get your package to you quickly and safely. All goods are new and include factory warranty. Due to our low prices all sales are final. **All defective returns must have a return authorization number.** Please call (414) 351-2007 to obtain an R.A. # or your return will not be accepted. Prices and availability subject to change without notice.

Hex is a colorful and deceptively simple game. Your goal is to change the 19 hexagonal blocks of the arena to green before your opponent—controlled by the computer—can turn them to purple. All it takes to change a block's color is to jump on it with your onscreen character. Each block changes color in a sequence displayed at the start of each level: green, then red, then purple, and finally blue before the sequence repeats. Seemingly a simple enough task.

Each of the 12 opponents you face employs a different strategy. Some opponents try to overpower you by the force of their magic, while others combine magic with cunning strategy and wisdom. If that isn't enough, at higher levels you may be confronted by several rival magicians, all working against you. You must learn the game well and plan your strategy carefully on the lower levels, because the computer becomes relentless as you progress to higher levels. There is no margin for error.

After successfully completing a round, you're offered the chance to learn a new magic spell. Each spell is costly, and you must exercise good judgment before undertaking the necessary instruction. Is the cost of the spell too high in energy points? Will it leave you too weak to face your next opponent?

Fast reflexes won't help in *Hex*. In fact, the key to early success is to play slowly and carefully, considering each move, much as in chess. Don't let the speed at which the computer plays trick you into making quick decisions. You're facing an opponent who is analyzing your tactics much faster than you can respond. Each rapid-fire move by the computer has a purpose; you must watch closely and try to disrupt its plans.

Hex may be one of the most challenging and fascinating strategy games yet devised for a computer. Although the game is simple to learn, you need to develop complex strategies to win consistently. You'll be amazed at how quickly your opponent ceases to be just a computer and seems to acquire distinct personality traits of its own.

Hex
Mark of the Unicorn, Inc.
222 Third Street
Cambridge, MA 02142
$39.95

Sylvia Porter's Personal Financial Planner

Selby Bateman, Features Editor

Requirements: Commodore 64 or 128 (in 64 mode or 128 mode); Apple IIc or IIe with 128K RAM; or an IBM PC/PCjr with at least 128K RAM. One or two disk drives are also required. Printer optional, but highly recommended. The Commodore 64 version was reviewed.

For many people, gaining control of a household budget is an exercise in frustration. Where do you start? How do you organize all those daily purchases, bill payments, unexpected expenses, and (far-too-few) paychecks into a coherent picture? Faced with this confusion, many of us go from day to day and month to month with little idea how much we have, how much we owe, and what's left over for savings and long-term financial goals. This is especially critical now, when consumer debt is at an all-time high and personal savings have plummeted.

The good news is that you *can* bring order to your financial chaos. *Sylvia Porter's Personal Financial Planner* is a well-organized, flexible, and sophisticated computer program that can make a major difference in your budgeting and planning efforts. The sobering news is that you're still going to have to invest a significant amount of time and concentration to set up your personal system and then use it on a regular basis. This isn't meant as a criticism of the *Personal Financial Planner*, however. It's simply a reality of personal financial planning in general, whether you manage it with pen and paper or on a computer screen.

Many people will be familiar with Sylvia Porter's name. She's been a respected and popular financial adviser for years—the author of a variety of articles and books, plus a nationally syndicated newspaper column, about budgeting, financial planning and management, and economics. More recently, she's lent her name and expertise to *Sylvia Porter's Personal Finance Magazine*.

The editors of that magazine have contributed to the overall approach and content of the *Personal Financial Planner*, which is supposed to be the first module in an integrated series of financial planning and management programs from Timeworks bearing Sylvia Porter's name. The next program, tentatively scheduled for this spring or summer, is *Personal Investment Planner*.

Six Programs In One

The strength of *Personal Financial Planner* lies in its flexibility, integration of information, and its well-planned structure. Think of *Personal Financial Planner* as six interrelated programs which share all of your financial information:

Transaction Manager: A program that lets you record and monitor all of your cash, bank account, and credit card transactions.

Budget Manager: A budget planning tool which automatically incorporates information from the Transaction Manager.

Asset/Liability Manager: An overview showing all that you own and all that you owe.

Income and Expense Statement: A part of the program that lets you organize and then print out income and expense statements in a variety of ways.

Balance Sheet: A similar component which allows you to arrange and print your asset and liability statements.

Financial Planner: A long-range planning guide that helps you set goals based on your income, expenses, and your changing asset/liability picture.

Pull-down menus and submenus make it quite easy to move around in the system. The documentation is clear, even for someone unfamiliar with computers.

Backups Take Time

Before you can begin using the program, you must initialize a data disk for each of the program managers—three data disks in all. On the Commodore 64, this initialization process requires more than a half-hour to complete. A data disk can generally store up to 1,250 transactions, so this initialization is only an occasional necessity. However, making backup copies of your data disks (an important precaution) is also time-consuming. The backup process doesn't just add new information to the backup disk; it completely rewrites the disk each time you make a backup. Because of the delay, it's tempting to skip this step now and then—risking disaster if your original disk should get lost or crash.

At least with the Commodore 64 version, there are a few instances when the manual doesn't mention that disk swaps are necessary. However,

onscreen prompts are very helpful here. And although the disk swapping can be an annoyance, the limitations lie with the 64 and 1541 disk drive, rather than with the program itself. Other computer versions, while functionally similar, have more space for information storage than the Commodore 64 version.

Once your data disks are prepared, your next step is to use the Transaction Manager to enter two-digit codes for up to five bank accounts (checking, money market, etc.) and up to ten credit card accounts, along with complete account information. As a part of this initial cataloging, you'll also set up a series of transaction/budget categories that you'll use with your various transactions. There are 14 major categories, including Income, Loans, Taxes, Groceries, Residence, Utilities, Clothing, Transportation, Insurance, Recreation, Medical/Dental, Education, Miscellaneous, and Other.

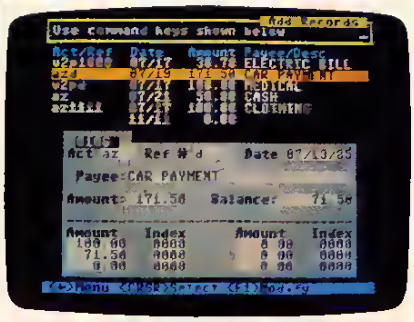
Each category has up to ten subcategories—a total of 140 separate budget/transaction items. What's more, each can be individually tailored to your specific requirements, a very nice feature of this program.

Why all of those categories and codes? If *Personal Financial Planner* was just a checkbook balancing program or a simple budget package, little cross-referencing would be required. But each of the categories you establish can be transferred among the Transaction, Budget, and Asset/Liability managers. Hence, the computer must have a good way of keeping track of each item. This is also important when you later want the program to find and print (on screen, paper, or disk) information on individual accounts, credit cards, or subcategories. Once you've set these up and used the program a couple of times, you'll find you're comfortable with the structure. And you can easily generate a printout of the different categories and codes for quick reference.

Calculator And Notepad

The initial effort it takes to establish budget categories is the most time-consuming aspect of *Personal Financial Planner*. Once that's done, much of the program transfers information automatically, or with just a few keystrokes. Templates automatically appear on the screen, letting you add, delete, and alter virtually any part of your budget. The program also lets you include information on automatic transactions—those recurring accounts such as rent, house payments, or loans—so that each month you don't have to enter all of the information by hand.

Among the many features are procedures for monthly reconciliation of



bank statements; searching for, changing, and printing out almost any part of your transaction, budget, or asset/liability data; automatically updating budget goals versus budget realities; setting up graphs and charts to show important aspects of your budget; tracking your financial inventory; and using financial planning worksheets that can be compared and contrasted with past, present, and future financial information.

There are many nice touches in *Personal Financial Planner*. In addition to the flexibility within each of the manager sections, there's a calculator and a notepad which can be called up at anytime. Also, you can search and modify your data, print out checks, track and print out tax information, and produce custom-tailored financial statements.

Timeworks and Sylvia Porter have created a serious tool with which individuals and families can track just about any aspect of their finances. But for the program to be truly useful, you'll have to make a commitment to keep your transaction information up to date. And that means spending as much as an hour per week (sometimes more, depending on what you're doing) working with the *Personal Financial Planner*.

If you devote this time to using the program, you'll have a clearer picture of your financial status than ever before; your budgeting will be tied in with your daily transactions; and you'll find yourself planning for the future with concrete information. For those who have trouble budgeting, tracking their transactions, and planning toward financial goals, *Sylvia Porter's Personal Financial Planner* can be an excellent investment.

Sylvia Porter's Personal Financial Planner
Timeworks, Inc.
444 Lake Cook Road
Deerfield, IL 60015
Commodore 64 version—$59.95
Commodore 128 version—$69.95
Apple IIc/IIe version—$99.95
IBM PC/PCjr version—$129.95

HOTWARE: Software Best Sellers

This Month	Last Month	Title	Publisher	Remarks	Systems: Apple	Atari	Commodore	IBM	Macintosh
Entertainment									
1.	1.	*F-15 Strike Eagle*	MicraPrase	Air combot simulotion	●	●	●	●	
2.	2.	*Karateka*	Brøderbund	Actlan karote game	●	●	●		
3.	5.	*Ultima IV*	Origin Systems, Inc	Fontosy gome	●	●	●	●	
4.		*Silent Service*	MicraProse	Submorine simulotion	●	●	●	●	
5.	4.	*Flight Simulatar II*	SubLogic	Aircraft simulation	●	●	●		
Education									
1.	1.	*Typing Tutar III*	Simon & Schuster	Typing instruction progrom	●		●	●	●
2.	2.	*Math Blasterl*	Davidson	Introductory moth progrom, ages 6-12	●	●	●	●	
3.	3.	*New Impraved MasterType*	Scorboraugh	Typing instruction progrom	●	●	●	●	●
4.	4.	*Music Canstructian Set*	Electronic Arts	Music camposltian program	●	●	●		
5.		*I Am The C-64*	Creotive/ Activisian	Introduction to the C-64			●		
Home Management									
1.	1.	*Print Shap*	Brøderbund	Do-it-yourself print shop	●	●	●		
2.	2.	*The Newsraam*	Springboord	Do-it-yaurself newspaper	●		●	●	
3		*Print Shap Graphics Library III*	Brøderbund	Upgraded grophics library	●	●	●		
4.	4	*Print Shop Graphics Library*	Brøderbund	100 oddltional grophics	●	●	●		
5.	5.	*Three-in-One Bundle*	Timeworks	Word processor			●		

SpeedCalc
For Atari

Kevin Martin and
Charles Brannon, Program Editor

In response to popular request, COMPUTE! *presents this professional-quality spreadsheet program for Atari 400, 800, XL, and XE computers with at least 48K RAM. Written completely in high-speed machine language,* Atari SpeedCalc *has all the important features you'd expect from a commercial spreadsheet program. In addition, its data files can be merged into text files created with the Atari* SpeedScript *word processor published last year in* COMPUTE!. *Atari* SpeedCalc *requires a disk drive, and a printer is optional but recommended.*

Have you ever planned a budget for your home or office? If so, you probably used some sort of worksheet divided into rows and columns. Perhaps you wrote the months of the year along the top of the sheet and listed categories for earnings and expenses along one side. After entering data for each category and month of the year, you could calculate total income figures by adding or subtracting numbers in each of the sheet's "cells."

That's a classic example of a worksheet. It lets you enter and organize data, then perform calculations that produce new information. A *spreadsheet* program is an electronic version of the familiar paper worksheet. Since it does all the calculations for you at lightning speed, an electronic spreadsheet is far more convenient than its paper counterpart. And spreadsheet programs also offer editing features that let you enter and manipulate large amounts of data with a minimum of effort.

Atari SpeedCalc is an all machine language spreadsheet program for Atari 400, 800, XL, and XE computers with at least 48K RAM. Though relatively compact in size, *SpeedCalc* is fast, easy to use, and has many of the features found in commercial spreadsheet programs.

Even better, if you print a *SpeedCalc* file to disk (see below), you can then merge it with a word processing document created with *SpeedScript*, COMPUTE!'s popular word processor (see COMPUTE!, May 1985, or *SpeedScript: The Word Processor for Atari Computers*, published by COMPUTE! Books).

Working together, *SpeedCalc* and *SpeedScript* make a powerful team. You can merge a chart of sales figures into a company report, create a table of scientific data for a term paper, and manipulate numeric information in many other ways. In a sense, a spreadsheet program brings to arithmetic all of the flexibility and power that a word processor brings to writing.

Preparing The Program

Although Atari *SpeedCalc* is small in comparison to similar commercial programs, it is one of the longest programs COMPUTE! has ever published. Fortunately, the "Atari MLX" machine language entry utility makes it easier to type a program of this size. Be sure to carefully read the Atari MLX article elsewhere in this issue before you begin. Here are the addresses you need to enter *SpeedCalc* with Atari MLX:

Starting address: 8192
Ending address: 16813
Run/Init Address: 8192

Next you'll be asked "Tape or Disk." *SpeedCalc* requires a disk drive, so type D. MLX will ask "Boot Disk or Binary File." Press F to select the Binary File option. (Although you could save *SpeedCalc* as a boot disk, it makes no sense, since such a disk cannot contain DOS, and DOS is necessary for file-oriented disk access.)

The screen then shows the first prompt, the number 8192 followed by a colon. Type in each three-digit number shown in the listing. You do not need to type the comma shown in the listing. MLX inserts the comma automatically.

The last number you enter in each line is a *checksum*. It represents the values of the other numbers in the line summed together. If you type the line correctly, the checksum calculated by MLX and displayed on the screen should match the checksum number in the listing. If it doesn't match, you have to retype the line. MLX is not foolproof, though. It's possible to fool the checksum by exchanging the position of the three-digit numbers. Also, an error in one number can be offset by an error in another. MLX will help catch your errors, but you still must be careful.

If you want to stop typing at some point and pick up later, press CTRL-S and follow the screen prompts. MLX will ask you for a disk filename; use any legal Atari filename except AUTORUN.SYS. Remember to note the line number of the last line you entered. When you are ready to continue typing, load MLX, answer the prompts as you did before, then press CTRL-L. For a binary disk file, MLX asks for the filename you gave to the partially typed listing. After the LOAD is complete, press CTRL-N and tell MLX the line number where you stopped. Now continue typing as before.

Saving The Finished Program

When you finish all typing, MLX automatically prompts you to save *SpeedCalc*. For disks with Atari DOS 2.0, 2.5, or 3.0, save the completed program with the filename AUTORUN.SYS. This allows *SpeedCalc* to load and run automatically when you boot the disk.

Because *SpeedCalc* requires a full 48K of RAM in order to work, you must *always* disable BASIC before loading or running *SpeedCalc*. On an Atari 400, 800, or 1200XL, unplug the BASIC cartridge (or any other cartridge, for that matter). On an Atari 600XL, 800XL, or 130XE, unplug any cartridges and disable BASIC by holding down the OPTION button when you first switch on and boot the computer. If you forget to disable BASIC, *SpeedCalc* won't work correctly.

To use *SpeedCalc* with an Atari DOS disk, you must save or copy it on a disk that also contains DOS.SYS and DUP.SYS. Since you've saved *SpeedCalc* as AUTORUN.SYS, it will automatically load and run when you turn on your computer with this disk in the drive. *SpeedCalc* should always be named AUTORUN.SYS in order to load properly with Atari DOS. If you want to prevent it from automatically running for some reason, you can save it with another name, then rename it AUTORUN.SYS later.

If you're using Optimized System Software's OS/A+ DOS or a compatible successor, you can give *SpeedCalc* any filename you like. Just use the LOAD command from DOS, and *SpeedCalc* will automatically run. Or you can give it a filename with the extension .COM, such as CALC.COM. Then you can start up by just typing CALC at the DOS prompt. You can also write a simple batch file to boot up *SpeedCalc* automatically. Some enhanced DOS packages may use so much memory that they conflict with *SpeedCalc*. In this case, you'll need to use Atari DOS instead on your *SpeedCalc* disks.

Note: The AUTORUN.SYS file on your DOS master disk is responsible for booting up the 850 Interface Module for RS-232 communications. There is no easy way to combine the 850 boot program with *SpeedCalc*, so you can't access the R: device while using this program. If you need to send a *SpeedCalc* file to a serial printer or modem, print it to disk as explained below, then print or transmit the file data as you would any ATASCII text.

The Atari *SpeedCalc* Screen

SpeedCalc uses the top line of the screen as the *command line*. This is where *SpeedCalc* displays messages and asks you questions.

Screen lines 2–4 are the *input buffer* area. This is the work area where you enter and edit data. As you'll see in a moment, the input buffer also displays the data contained in the current cell. The work area cursor is a left arrow symbol (←). After you begin to enter data, most *SpeedCalc* commands (except for the cursor movement keys) are deactivated until you press RETURN to enter the data into the worksheet.

The lower 19 screen lines are your window into the spreadsheet. Though the spreadsheet contains many rows and columns, only a few can fit on the screen at one time. By scrolling the screen back and forth with the cursor, you can

From the publishers of *COMPUTE!*

March 1986 *COMPUTE!* Disk

All the exciting programs from the past three issues of *COMPUTE!* are on one timesaving, error-free floppy disk that is ready to load on your Atari 400/800, XL, and XE. The March 1986 *COMPUTE! Disk* contains the entertaining and useful Atari programs from the January, February, and March 1986 issues of *COMPUTE!*. This easy-to-use disk also features *SpeedCalc*, the spectacular new spreadsheet program written entirely in machine language for the Atari, and the latest version of *SpeedScript*, the bestselling word processing program.

The March 1986 *COMPUTE! Disk* costs $12.95 plus $2.00 shipping and handling and is available only from COMPUTE! Publications.

For added savings and convenience, you may also subscribe to the *COMPUTE! Disk*. At a cost of only $39.95 a year (a $12.00 savings), you'll receive four disks, one every three months. Each disk will contain all the programs for your machine from the previous three issues of *COMPUTE!*.

This is an excellent way to build your software library while you enjoy the quality programs from *COMPUTE!*.

Disks and subscriptions are available for Apple, Atari, Commodore, and IBM personal computers. Call for details.

For more information or to order the March 1986 *COMPUTE! Disk*, call toll free 1-800-247-5470 (in Iowa 1-800-532-1272) or write *COMPUTE! Disk*, P.O. Box 10036, Des Moines, IA 50340.

COMPUTE! Publications, Inc. abc
One of the ABC Publishing Companies

move the display window to any part of the spreadsheet.

The *SpeedCalc* worksheet consists of 50 vertical columns labeled with letters (AA, AB ... BX) and 100 horizontal rows numbered from 1–100. The rectangle where a row and column intersect is called a *cell*. Cells are where you store data. With 50 columns and 100 rows, the *SpeedCalc* spreadsheet has a maximum of 5,000 (50*100) cells. Due to memory limitations, however, only about a third of these can actually contain data. But you may spread out the data over all 5,000 cells if necessary, depending on the format you need.

Moving The Cursor

Each cell is identified with the letters of its column and the number of its row. For example, the cell at the extreme upper-left corner of the sheet is called AA1, since it's in column AA and row 1. The cell below that is AA2. Moving one cell to the right from AA2 puts you in cell AB2, and so on.

Your current position in the spreadsheet is shown by the highlighted cursor. The simplest way to move around the sheet is with the cursor keys, which work just as they do when you're writing or editing a BASIC program. Press CTRL and the right cursor key to move right, and so on. Another way to move the cursor is with CTRL-H. Press CTRL-H once to "home" the cursor on the current screen: The cursor moves to the upper-left cell. Press CTRL-H twice in succession to move the cursor to cell AA1, the home position for the entire sheet.

SpeedCalc also has a *goto* command for moving the cursor over long distances. When you press CTRL-G, the command line displays GOTO: followed by a cursor. The cursor generally indicates that *SpeedCalc* is waiting for data—in this case it expects the name of the cell where you wish to go. If you enter BA88 at this point, *SpeedCalc* moves the cursor to the cell at column BA in row 88, adjusting the screen window as needed. Take a few moments to practice moving around the spreadsheet with all three methods; you'll be using them a lot. In a later section, we'll discuss how to change the size and format of a cell.

Keyboard Commands

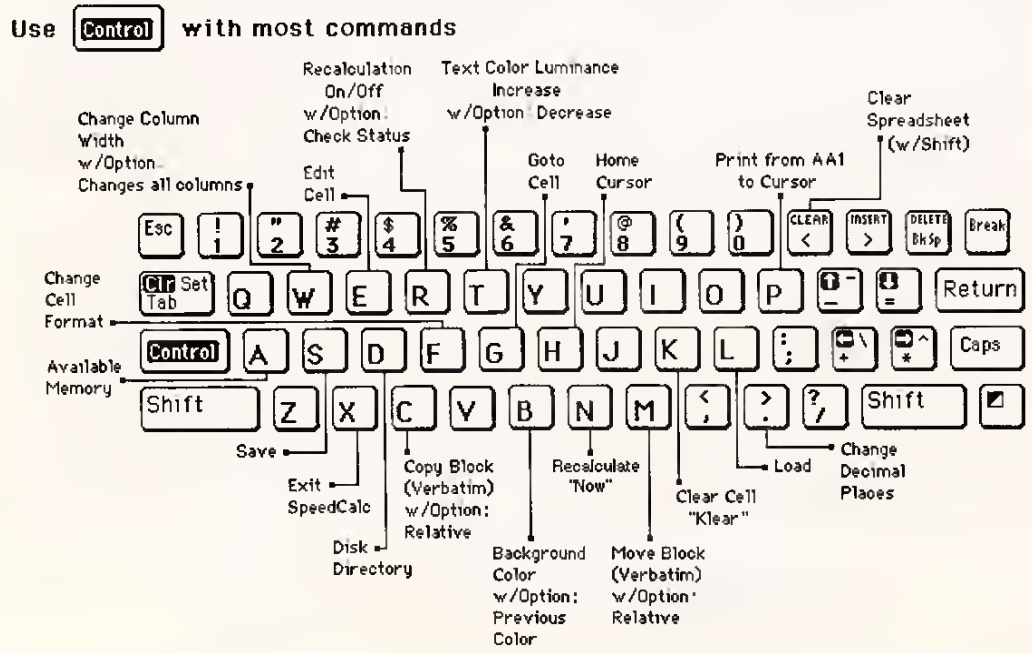
SpeedCalc offers many different commands, a few of which are entered by pressing one key. However, most commands are entered by pressing CTRL along with another key. CTRL-G, as you've seen, is the goto command. CTRL-A displays the amount of free memory available, and so on.

The most drastic command is CTRL-X, which exits *SpeedCalc* and returns to DOS. Since this effectively erases all data in memory, *SpeedCalc* prompts you with ARE YOU SURE Y/N? before it shuts down. To cancel the command, simply type N (or any key other than Y). If your Atari DOS 2.0/2.5 disk contains the file MEM.SAV (created with the CREATE MEM.SAV option on the DOS menu), you can exit to DOS and then return to *SpeedCalc*—however, all spreadsheet data will be lost. To restart *SpeedCalc* from the DOS menu after exiting, select menu option M (Run At Address), then enter the address 2000. If you're using OS/A+ or DOS XL, use RUN 2000 instead.

If you press SYSTEM RESET in *SpeedCalc*, you'll see the message SYSTEM RESET TRAPPED. No spreadsheet data is lost. If you're using OS/A+ or DOS XL, type RUN 2000 to return to *SpeedCalc*.

A few commands require you to press three keys at once. This sounds more awkward than it is in practice, since two of the three keys are OPTION and CTRL. For instance, the *relative copy* command

SpeedCalc Keyboard Reference



is performed by pressing OPTION-CTRL-C (hold down the OPTION console key and CTRL, then press C). The table lists all the *SpeedCalc* commands, and the figure shows the keyboard layout with a description of what each key does. We'll be discussing each command in more detail below.

Three Data Types

Before entering any data, you must know what kind of data *SpeedCalc* accepts. There are three different types: numbers, text, and formulas. Let's look at each type in turn.

1. Numeric data consists of numbers—the basic stuff that spreadsheets work with. *SpeedCalc* has a few simple rules for numeric data: A number must be a decimal value (base 10, not hexadecimal) composed of one or more digits from 0–9, with an optional plus or minus sign. A decimal point is also optional. If you include any other characters in numeric input, *SpeedCalc* treats the entire input as text data (as explained below). Thus, the numbers 123, .001, and −65535 are valid numeric data. The number 65,535 is invalid because it includes a comma.

The allowable range for numbers in Atari *SpeedCalc* is similar to the range for Atari BASIC, roughly −1.7E+97 to 1.7E+98. If a calculation produces a number outside the allowable range, you'll see the message *ERROR* in the cell containing the formula. This doesn't happen very often, since *SpeedCalc* won't let you enter a number more than 36 digits long, and there's rarely a need to use such large numbers unless you're tracking the national debt.

Although an input value can be up to 36 digits long, numbers in *SpeedCalc* calculations are accurate only to nine digits. This must be taken into account when doing any calculation involving large values. For example, you can enter the value 1122334455.66 into a cell, and the cell holds the value with no rounding. However, if you use the value from that cell in a formula, the value is rounded to nine digits—112233446.00—and the result of the calculation is accurate only for the first nine digits.

You can enter values in scientific notation by following a number with the letter E and the appropriate power of 10. For example, you can enter 1,234,000 as 1.234E+06. However, scientific notation should generally be avoided, since values outside the Atari's maximum range may crash the program (if this happens, press RESET and rerun the program from DOS as explained above). Since there's only room for about 36 digits, unpredictable results may occur if you enter any number in scientific notation with an exponent greater than 35 (E+35).

To see how entering numeric data works, let's enter the number 123 in cell AA1. No special commands are required to enter data: Just move the cursor to AA1 and begin typing. The left-arrow symbol shows the end of the data. While you're entering the number, it appears only in the input buffer near the top of the screen (the inverse-arrow cursor shows your cursor position). As soon as you press RETURN, the number appears in cell AA1 and the letter N appears at the upper right of the screen. The N signifies *numeric*, meaning that *SpeedCalc* has accepted the entry as valid numeric data. Move the cursor to a vacant cell, then move it back to AA1. The input buffer displays whatever data is found in the cell under the cursor. When the current cell is empty, the buffer is empty as well.

If you want to change anything during data entry, press the BACKSPACE key (BACKS on some Atari machines). BACKSPACE always deletes the character before the cursor (or has no effect if the cell is empty). Later on, we'll explain how to edit existing data.

As you've seen, pressing RETURN enters a data item into the current cell. You can also end the input by pressing CTRL and a cursor key. The data is entered as if you had pressed RETURN, and the cursor moves in the indicated direction. This feature is handy for entering a lot of data: Simply type the entry, move the cursor to the next cell, enter more data, and so on.

2. Text data is not "data" in the strict sense, since *SpeedCalc* doesn't use it in calculations as it does numbers and formulas. Text data is there only to help people understand what the other data means. Text may consist of comments, titles, column headings, subheadings, or whatever you need to interpret the numbers and formulas. As an example, move the cursor to cell AA2 (just under AA1) and type the following line:

THIS IS A PIECE OF TEXT DATA.

You can use the BACKSPACE key to erase mistakes while you're typing. When you press RETURN, *SpeedCalc* displays T (for text) in the upper-right corner. In this example, the cell isn't large enough to accept all the text, so only the leftmost portion appears in AA2. But even though you can't see it, all of the text is there. Move the cursor to another cell, then move it back to AA2. As soon as you return to AA2, *SpeedCalc* displays all the text in the input buffer area.

3. Formula data is a mathematical expression or formula. It may be as simple as 2 + 2 or as complex as your imagination (and mathematical prowess) allows. The first character in a formula must always be an equal sign (=). If you omit this symbol, *SpeedCalc* either signals an error or treats the data as text.

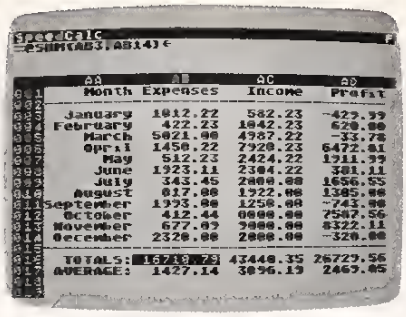
The true power of a spreadsheet is that a formula in one cell can refer to another cell. This is easier to demonstrate than to explain. Move the cursor to cell AA3 and type the following line:

=AA1*25.01+@SQR(4)

As soon as you press RETURN, *SpeedCalc* displays F (for formula) in the upper-right corner of the screen and puts the *result* of the formula (not the formula itself) in AA3. If AA1 contains 123, the value 3078.23 appears in AA3. In plain English, this formula means "multiply the contents of cell AA1 by 25.01 and add the square root of 4."

Before we examine the formula more closely, here's a quick demonstration of what makes a spreadsheet such a powerful tool. Move the cursor back to AA1 and press CTRL-R. The command line displays the message RECALCULATION IS ON, meaning *SpeedCalc* now automatically recalculates the

A typical screen from Atari SpeedCalc*—a compact, powerful spreadsheet program written entirely in machine language.*



Atari SpeedCalc*'s input buffer always displays the contents of the data cell under the highlighted cursor.*

entire sheet whenever you make a change. Now change the number in AA1 to 456 (simply move to the cell and start typing). The new result (11406.56) automatically appears in cell AA3. We'll explain more about automatic recalculation later.

Note that the referenced cell must contain data that *SpeedCalc* can evaluate: a number or another formula. If the formula refers to an empty cell, or one that contains text, *SpeedCalc* signals the error by printing *ERROR* in the cell containing the incorrect formula.

Mathematical Operators

These symbols can be used as *operators* in a formula:

Operator	Function
+	addition
−	subtraction
*	multiplication
/	division
^	exponentiation
=	equality

One factor that affects formulas is *precedence*, or the order in which mathematical operations are performed. In *SpeedCalc*, formula operators have the same precedence as in ordinary math.

The first operators to be evaluated—those with the highest precedence—are those enclosed in parentheses. Where one set of parentheses encloses another, the expression in the innermost set is evaluated first. The next operators to be evaluated are exponents. Multiplication and division have equal precedence; both operations are lower than exponentiation. Addition and subtraction have the lowest precedence of all. To take one example, *SpeedCalc* evaluates the formula =5*(8+3*−2)^2−10/+2 as the value 15, just as in ordinary math. Note how the result is affected by the plus and minus signs before the two 2's.

Functions

Formulas may also include any of the functions listed here:

@ABS()	absolute value
@AVE()	average of a block of cells
@EXP()	natural exponent
@INT()	integer
@LOG()	natural logarithm
@RND()	round to nearest integer
@SGN()	sign
@SQR()	square root
@SUM()	sum of a block of cells
PI	value of pi (3.14159265)

All the functions except PI begin with the @ symbol and are followed by parentheses. The parentheses of a function may contain a number or a formula. For example, the formula =@SQR(4) generates the square root of 4. The formula =@SQR(AA1) returns the square root of whatever value cell AA1 contains. The function @INT() generates an integer (whole number) by truncating (discarding the fractional part of) a numeric value; note that this is different from rounding (for instance, the result of @INT(−4.3) is −4, not −5). Use the rounding function @RND() to round a value up to the nearest whole number.

The function @AVE() calculates the mean average of the values in a block (group) of cells. The function @SUM() calculates the sum of a block. Both functions require you to define the block so *SpeedCalc* knows which cells to include in the calculation. This is done by putting two cell names separated by a colon in the parentheses. The first cell name defines the upper-left corner of the block, and the second defines the bottom-right corner. To define a block in a single column, specify the top and bottom cells in the column. For instance, @AVE(AA1:AD20) calculates the average of all the cells from AA1 to AD20. The function @SUM (AA1:AD20) calculates the sum of AA1 through AD20, and so on. *An error results if any cell in the block is blank or contains text data.*

Editing The Sheet

Editing is a very important spreadsheet function. The simplest way to change what a cell contains is to move to it and start typing. The old data in that cell is replaced by whatever you enter. For instance, to replace the contents of cell AA1 with the number 456, move to that cell, type 456, and press RETURN or exit with a cursor key. Press CTRL-K (think of *kill*) to erase what's in the current cell. To erase everything in the sheet, press SHIFT-CLEAR. Before carrying out this drastic operation, *SpeedCalc* asks you to confirm it by pressing Y or N.

In some cases, only a minor change is needed. *Edit mode* lets you change the data in a cell without retyping the entire entry. To activate edit mode, move to the desired cell and press CTRL-E. In this mode, up and down cursor movement is disabled, and the left/right cursor keys move within the input buffer. Typing in edit mode inserts new characters in the line: Everything to the right of the new character moves right one space (unless the buffer is already full). Because all keys insert automatically, the CTRL-INSERT key combination is disabled: Press the space bar to insert a blank space. Erase unwanted characters with the BACKSPACE key or CTRL-DELETE. The CTRL-DELETE combination does not move the cursor: It simply pulls the text to the right of the cursor toward the cursor position. Since the cursor keys have a different function in edit mode, you cannot use them to

end the input. Press RETURN to enter the new data and escape from edit mode.

SpeedCalc displays *ERROR* in a cell when you enter an erroneous formula. Usually this means you've made a typing error in that cell, or the formula refers to text or an empty cell. A line of asterisks (**********) signals that a number is too large to be printed in the cell. Though these messages appear in the cell area, no data is lost. You may move to the affected cell, view its contents in the input buffer, and make whatever correction is needed.

Recalculation

Recalculation is the very core of a spreadsheet. As you know, entering or editing a piece of data makes *SpeedCalc* perform a calculation and put the result in the cell under the cursor. In most cases, the new data relates to data in other cells, so you'll ultimately want to recalculate the entire spreadsheet as well. This can be done manually or automatically.

To recalculate the spreadsheet manually, enter CTRL-N. *SpeedCalc* begins at AA1 and recalculates every cell that contains data, placing fresh results wherever needed. If you switch to automatic recalculation mode, *SpeedCalc* automatically recalculates the entire spreadsheet each time you enter new data or edit what exists. When you press CTRL-R, *SpeedCalc* changes the recalculation status and displays it at the top of the screen. If automatic recalculation was turned off before, it is now on (and vice versa). If you aren't sure which mode you're in, press OPTION-CTRL-R; *SpeedCalc* displays the mode without changing it.

Automatic recalculation can be fun to watch in a large spreadsheet: Every time you make a change, new results appear everywhere on the screen. However, the more data your spreadsheet contains, the longer it takes to update the entire sheet. For this reason, you may want to turn off automatic recalculation most of the time, recalculating manually whenever you need to view results.

One problem with recalculation arises from the order in which cells are calculated. Because only one cell can be calculated at a time, you must sometimes recalculate the entire spreadsheet two or three times to get correct results in every cell (this is common to all spreadsheet programs). For instance, say you have a formula in AA1 which refers to a formula in AB15. When *SpeedCalc* calculates AA1, it must use the existing data from AB15—which is probably out of date, since the formula in AB15 hasn't been recalculated yet. To avoid this problem, you should always recalculate a sheet manually two or three times before printing it or saving it to disk.

SpeedCalc offers a number of other features. Before experimenting with them, you should spend some time typing in a hypothetical spreadsheet—perhaps a fictitious yearly budget—to become thoroughly familiar with the basic commands covered so far. Most importantly, create formulas using all the operators in different combinations.

Change Format

The default (normal) format for numeric data is flush right with rounding to two decimal places. In other words, the number is displayed in the rightmost part of the cell, with two numbers after the decimal point. Text and formulas are also displayed flush right. *SpeedCalc* offers several commands for changing cell formats.

Change Format (CTRL-F). This command changes the location of data in the cell. When you press CTRL-F, the *SpeedCalc* command line displays the question FORMAT: LEFT, CENTER, OR RIGHT JUSTIFY?. Press L, C, or R to move the data to the left, center, or right of the cell.

Change Decimal Places (CTRL-.). *SpeedCalc* also lets you change the number of decimal places for any cell. The default number of decimal places is 2, but you may change it to anything from 0–15. Press CTRL and the period key (CTRL-.) to change this value: *SpeedCalc* prompts you to enter a number from 0–15. If you choose zero decimal places, any number in that cell is rounded off to the nearest integer (whole number). If you choose 15, a number in that cell is not rounded off at all—*SpeedCalc* displays it exactly as you entered it or as it was calculated from a formula.

Width (CTRL-W). The width command changes the width of an entire column of cells. Move the cursor to any cell in the desired column, then press CTRL-W. When *SpeedCalc* displays the prompt WIDTH:, respond with a number from 4–36. The entire screen is redrawn to accommodate the new format, and may look very different depending on what value you chose. For instance, if you increase a column's width, the rightmost column of the former display may disappear: *SpeedCalc* only displays as many complete columns as it can fit on the screen. If you decrease the width of a column, you may see asterisks where numbers used to be (indicating the cell is now too small to display the entire number). To get rid of the asterisks, expand the column as necessary.

Global Format (OPTION-CTRL-F). This is the same as the ordinary format command, but operates globally, changing every cell in the sheet instead of just one.

Global Width (OPTION-CTRL-W). This is a global version of the width command. Every column in the sheet changes to the designated width.

Screen Color And Luminance

SpeedCalc makes it easy to change the screen background and character colors to your liking.

Background Color (CTRL-B). Press CTRL-B to cycle forward through the available screen background colors.

Text Color (CTRL-T). This command increases the luminance of characters on the screen, cycling forward through all of the available text colors.

Previous Background Color (OPTION-CTRL-B). The reverse of CTRL-B, this command cycles backward through the range of background colors.

Previous Text Color (OPTION-CTRL-T). The reverse of CTRL-T, this command cycles backward through the range of text colors.

Macro Editing

After typing in a large spreadsheet, you may decide to make a major change. You may want to add new data somewhere in the middle, delete a section, or move a group of cells from one location to another. *SpeedCalc*'s macro (large-scale) editing commands simplify such operations, affecting an entire block of cells at once. A *block* is simply a group of cells connected in rectangular fashion. You can define it as a single cell, a row or column, or any rectangular area within the spreadsheet.

There are two ways macro commands work: *verbatim* or *relative*. To take a simple example, say that cell AA2 contains the formula =AA1*5 and you want to move its contents to cell AB2. When this is done in verbatim mode, AB2 contains an exact copy of what was in AA2 (=AA1*5). Note that the cell name used in the formula does not change: The formula still refers to AA1. If you perform the same operation in relative mode, the cell name in the formula is adjusted to fit the new location. In this case, AB2 would contain the formula =AB1*5.

Copy (CTRL-C). The copy command copies a block of cells into a different location without disturbing the original cells. Place the cursor on the upper-left corner of the block you want to copy, then press CTRL-C. *SpeedCalc* prompts you to move the cursor to the lower-right corner of the block you want to copy. Once the cursor is in place, press RETURN. Now *SpeedCalc* prompts you to move the cursor to the place where you want to put the block: This is the upper-left corner of the new position. Once the cursor is there, press RETURN again. The new data replaces whatever was contained in the designated cells. Note that if you define an impossible block (for instance, moving the cursor to the upper-left of the original position, rather than below and to the right), *SpeedCalc* does not copy any data. Press ESC if you change your mind and wish to cancel this command.

Move (CTRL-M). This command works like a copy, but it fills the original cells with blanks. Though *SpeedCalc* has no express insert command, you can use this command to make space for new data in the middle of a spreadsheet. Simply move everything below the insertion point down as far as you need. As with the copy command, you can press ESC to cancel this command.

Relative Copy (OPTION-CTRL-C). This form of the copy command adjusts the cell names used in formulas within the copied block (see explanation above). When copying or moving data in relative mode, you may see some strange characters displayed very briefly in the input buffer area of the screen: This harmless effect occurs because *SpeedCalc* uses that area for temporary storage during these operations, conserving memory for other purposes.

Relative Move (OPTION-CTRL-M). This is the relative form of the move command. Cell names in formulas are adjusted to reflect the move.

Memory Management

SpeedCalc makes about 20K (roughly 20,000 characters) of memory available for data. As noted earlier, *SpeedCalc* lets you spread your data out over a much larger number of cells than you can actually fill with data. The extra space is provided to give you full control over the final format of the spreadsheet and to leave some elbow room for move and copy operations.

Because memory is limited, you should keep careful track of how much is free while using the program. Press CTRL-A to display the amount of free memory. We suggest limiting your spreadsheets to 1,600 cells (equivalent to 40 rows by 40 columns). If you've filled nearly all of free memory, you may have to break the spreadsheet into two smaller sheets.

Although *SpeedCalc* checks the amount of available memory and displays an error message if you run out, you should be careful not to exhaust free memory. Any move or copy operation in process will be aborted if sufficient memory is not available.

Disk Operations

SpeedCalc has three disk commands for saving and loading data from

SpeedCalc Commands

Command	Action
CTRL-A	available memory check
CTRL-B	next background color
CTRL-C	copy block verbatim
CTRL-D	disk directory
CTRL-E	edit current cell
CTRL-F	change cell format
CTRL-G	goto selected cell
CTRL-H	home cursor
CTRL-K	clear current cell
CTRL-L	load *SpeedCalc file*
CTRL-M	move block verbatim
CTRL-N	recalculate sheet now
CTRL-P	print cells from AA1 to cursor
CTRL-R	turn recalculation on/off
CTRL-S	save *SpeedCalc* file
CTRL-T	increase text luminance
CTRL-W	change column width
CTRL-X	exit *SpeedCalc* to DOS
CTRL-.	change decimal places
SHIFT-CLR	clear spreadsheet
OPTION-CTRL-B	previous background color
OPTION-CTRL-C	copy block relative
OPTION-CTRL-M	move block relative
OPTION-CTRL-R	check recalculation status
OPTION-CTRL-T	decrease text luminance
OPTION-CTRL-W	change width of all columns

disk and displaying the disk directory. The disk directory command is the easiest to use: Simply press CTRL-D. To save a spreadsheet to disk, press CTRL-S. *SpeedCalc* prints SAVE: on the command line, followed by a cursor. Enter a valid Atari filename (including D:) and press RETURN. (If you change your mind and decide not to save anything, press RETURN without typing a filename.) If no disk error occurs while the spreadsheet is being saved, *SpeedCalc* displays NO ERRORS in the command line and returns you to command mode. If there was an error, you'll hear a beep and see the message I/O ERROR # followed by an error number in the command line. Your DOS manual explains the meaning of the various DOS errors.

To load a saved file from disk, press CTRL-L. Again, you can cancel the operation by pressing RETURN without entering a filename. *SpeedCalc* prompts you to enter the filename and displays the error status when the operation is complete. If an error occurs while loading, *SpeedCalc* clears the partially loaded sheet to prevent a program crash.

Printing

SpeedCalc lets you print data to three different devices: to the screen for previewing output (E:), to a printer for permanent documentation (P:), or to a disk file for integrating the data with a *SpeedScript* document (D:*filename*).

To print a hardcopy of the spreadsheet to a printer, press CTRL-P and then enter P: when asked for (Device:Filename). *Before using this command, you must position the cursor below and to the right of the block of cells you wish to print.* The upper-left corner of the printout starts at cell AA1. To preview the printed output on the screen, enter E: in response to the same prompt.

You can also print *SpeedCalc* data to a disk file for use in a *SpeedScript* document. When *SpeedCalc* prints the prompt (Device:Filename), enter D:*filename*. The data is saved as a disk file of that name. Note that *printing* to disk creates a different type of file than *saving* to disk, and *SpeedCalc* cannot reload files in the print format. You should *save* files you wish to reload into *SpeedCalc*, and *print* files you wish to load into *SpeedScript*.

SpeedScript Integration

SpeedCalc sends data to the printer in simple, plain vanilla form. That may be fine for personal use, but if you're creating a document for others to view, you may want special features such as boldface, underlining, italics, and so on. Since Atari *SpeedScript*—COMPUTE!'s popular word processor—already offers a way to access these features (and many more), no attempt has been made to duplicate them in *SpeedCalc*.

No special tricks are needed to load a *SpeedCalc* file into *SpeedScript*. After printing the file to disk as explained above, exit *SpeedCalc*, then load and run *SpeedScript*. Now load the file as you would any *SpeedScript* document. The data appears on the screen, ready to be edited in any way you wish. Again, keep in mind that *SpeedScript* can load only those files which have been *printed* to disk, not saved.

Program 1: Atari *SpeedCalc*

Please refer to the "MLX" article in this issue before entering the following listing.

```
8192:165,089,201,188,240,018,133
8198:169,063,160,094,032,089,101
8204:033,032,028,033,169,001,052
8210:141,068,002,108,252,255,076
8216:169,063,160,035,162,000,101
8222:032,089,033,032,181,035,176
8228:032,026,035,169,066,024,132
8234:105,001,141,174,065,024,040
8240:105,041,133,159,169,000,143
8246:141,173,065,141,175,065,046
8252:133,158,141,049,062,169,004
8258:187,141,176,065,169,203,239
8264:205,054,066,141,054,066,146
8270:240,030,032,108,035,165,176
8276:012,141,005,033,165,013,197
8282:141,006,033,169,004,133,064
8288:012,169,033,133,013,169,113
8294:000,141,068,002,169,001,227
8300:133,009,032,182,035,032,019
8306:003,038,032,028,033,072,064
8312:032,152,033,104,174,173,020
8318:032,221,173,032,240,022,078
8324:202,208,248,201,032,144,143
8330:230,201,123,176,226,201,015
8336:091,144,004,201,097,144,057
8342:218,076,182,036,202,138,234
8348:010,170,169,032,072,169,010
8354:112,072,189,213,032,072,084
8360:189,212,032,072,096,024,025
8366:125,008,023,006,007,016,103
8372:003,019,012,024,028,029,039
8378:031,030,011,005,014,002,023
8384:020,001,018,004,013,096,088
8390:013,017,018,019,020,021,050
8396:022,023,024,025,016,011,069
8402:013,014,083,035,067,042,208
8408:127,040,041,037,178,041,168
8414:117,044,244,045,217,049,170
8420:179,050,084,054,090,041,214
8426:065,041,113,041,155,041,178
8432:035,052,096,052,149,051,163
8438:070,033,052,033,131,053,106
8444:054,052,015,058,212,045,176
8450:107,037,032,064,021,032,039
8456:026,035,032,003,038,032,174
8462:182,035,169,063,160,072,183
8468:162,007,032,089,033,076,163
8474:113,032,173,252,002,201,031
8480:255,240,249,133,146,138,169
8486:072,152,072,032,125,059,038
8492:133,151,104,168,104,170,106
8498:165,151,096,162,002,032,146
8504:109,058,208,002,162,254,081
8510:024,138,109,052,062,141,076
8516:052,062,096,162,002,032,218
8522:109,058,208,002,162,254,099
8528:024,138,109,051,062,141,093
8534:051,062,096,133,204,132,252
8540:203,142,050,062,169,000,206
8546:133,084,133,085,169,001,191
8552:141,240,002,032,140,033,180
8558:160,000,140,255,002,177,076
8564:203,240,006,032,229,058,116
8570:200,208,246,096,162,050,060
8576:157,180,065,202,208,250,166
8582:169,040,141,231,065,096,108
8588:160,000,169,000,145,088,190
8594:200,192,040,208,247,096,105
8600:173,065,188,201,112,240,107
8606:009,169,063,160,025,162,234
8612:000,032,089,033,056,032,150
8618:078,055,144,003,076,165,179
8624:039,076,175,039,032,176,201
8630:034,141,104,188,169,094,144
8636:141,105,188,162,118,169,047
8642:000,157,105,188,202,208,030
8648:248,160,001,208,002,160,211
8654:000,185,104,188,009,128,052
8660:153,104,188,032,028,033,238
8666:141,249,065,185,104,188,126
8672:041,127,153,104,188,173,242
8678:249,065,174,154,034,221,103
8684:154,034,240,055,202,208,105
8690:248,201,032,144,216,201,004
8696:125,176,212,032,176,034,235
8702:141,249,065,140,250,065,140
8708:206,250,065,162,119,189,227
8714:104,188,201,094,240,191,004
8720:202,189,104,188,157,105,193
8726:188,202,236,250,065,208,147
8732:244,173,249,065,153,104,248
8738:188,200,076,207,033,202,172
8744:138,010,170,189,163,034,232
8750:072,189,162,034,072,096,159
8756:160,000,185,104,188,201,122
8762:094,240,006,153,000,006,045
8768:200,208,243,169,000,153,013
8774:000,006,140,234,065,096,099
8780:173,054,062,240,032,192,061
8786:000,240,001,136,076,207,230
8792:033,173,054,062,240,019,157
8798:185,104,188,201,094,240,082
8804:241,200,076,207,033,173,006
8810:054,062,240,003,076,207,236
8816:033,165,146,141,252,002,083
8822:076,052,034,192,000,240,200
8828:217,136,185,104,188,201,131
8834:094,240,209,152,170,189,160
8840:105,188,157,104,188,232,086
8846:201,094,208,245,169,000,035
8852:157,104,188,076,207,033,145
8858:007,155,126,028,029,030,017
8864:031,254,051,034,120,034,172
8870:104,034,104,034,075,034,039
8876:088,034,125,034,072,041,054
8882:128,133,151,104,041,127,094
8888:201,096,176,013,201,032,135
8894:176,006,024,105,064,076,129
8900:201,034,056,233,032,005,245
8906:151,096,072,041,128,133,055
8912:151,104,041,127,201,096,160
8918:176,011,201,064,144,005,047
8924:233,064,076,227,034,105,191
8930:032,005,151,096,072,138,208
8936:072,173,051,062,069,079,226
8942:037,070,141,010,212,141,089
8948:024,208,141,200,002,173,224
8954:052,062,069,079,037,078,115
8960:141,023,208,174,050,062,146
8966:189,055,062,141,198,002,141
8972:169,010,141,197,002,169,188
8978:000,141,182,002,104,170,105
8984:104,064,169,064,141,014,068
8990:212,169,230,141,000,002,016
8996:169,034,141,001,002,173,044
9002:048,002,133,151,173,049,086
9008:002,133,152,160,003,169,155
9014:194,145,151,169,192,141,022
9020:014,212,096,169,064,141,244
9026:014,212,173,051,062,141,207
```

```
9032:198,002,141,200,002,173,020
9036:052,062,141,197,002,096,116
9044:169,063,160,226,162,002,098
9050:032,089,033,032,028,033,081
9056:041,095,201,089,208,003,221
9062:032,108,035,076,152,033,026
9068:032,141,035,169,009,032,014
9074:126,033,169,044,141,144,003
9080:062,032,182,035,032,003,210
9086:038,165,158,133,136,165,153
9092:159,133,137,169,000,141,103
9090:143,062,096,173,173,065,082
9104:133,203,173,174,065,133,001
9110:204,160,000,152,145,203,246
9116:200,208,251,230,204,166,135
9122:204,236,176,065,208,242,013
9128:169,001,141,178,065,141,095
9134:179,065,133,134,133,135,185
9140:096,096,032,188,035,076,191
9146:050,036,165,088,024,105,142
9152:200,133,142,165,089,105,002
9158:000,133,143,160,000,174,040
9164:179,065,169,000,133,151,133
9170:133,152,248,165,151,024,059
9176:105,001,133,151,165,152,155
9102:105,000,133,152,202,208,254
9188:240,216,162,000,032,018,128
9194:036,248,165,151,024,105,195
9200:001,133,151,165,152,105,179
9206:000,133,152,216,165,142,030
9212:024,105,040,133,142,165,093
9218:143,105,000,133,143,160,174
9224:000,232,224,019,208,210,141
9230:032,010,036,096,165,152,001
9236:024,105,144,145,142,200,012
9242:165,151,041,240,074,074,003
9248:074,074,024,105,144,145,086
9254:142,200,165,151,041,015,240
9260:024,105,144,145,142,096,188
9266:024,165,088,105,160,133,213
9272:142,165,089,105,000,133,178
9278:143,160,000,169,128,145,039
9284:142,200,145,142,200,145,018
9290:142,200,174,178,065,169,234
9296:000,141,177,065,189,180,064
9302:065,134,151,074,105,000,103
9300:170,202,169,128,145,142,024
9314:200,202,208,250,165,151,250
9320:010,170,109,145,062,009,177
9326:128,145,142,200,189,146,036
9332:062,009,128,145,142,200,034
9338:166,151,189,180,065,074,179
9344:170,202,202,169,128,145,120
9350:142,200,202,016,250,166,086
9356:151,189,180,065,024,109,090
9362:177,065,141,177,065,232,235
9368:189,180,065,024,109,177,128
9374:065,201,037,144,177,202,216
9380:142,240,065,169,128,192,076
9386:040,208,001,096,145,142,034
9392:200,192,040,208,249,096,137
9398:032,180,033,173,000,006,094
9404:240,063,201,029,240,038,231
9410:174,198,032,221,198,032,025
9416:240,007,202,208,248,169,250
9422:001,208,025,173,234,065,144
9428:201,037,176,037,160,000,055
9434:169,006,032,000,037,032,238
9440:028,062,208,233,169,000,156
9446:240,002,169,002,141,232,248
9452:065,173,144,062,141,235,032
9458:065,024,032,078,055,032,016
9464:174,055,032,144,051,076,012
9470:113,032,141,030,062,140,004
9476:029,062,032,028,062,076,037
9482:221,060,162,050,169,000,160
9488:141,246,065,189,180,065,134
9494:024,109,246,065,141,246,085
9500:065,201,037,176,003,202,200
9506:208,239,232,232,142,250,057
9512:065,076,032,109,058,240,128
9518:004,162,006,208,002,162,078
9524:014,160,137,169,064,032,116
9530:089,033,032,028,033,041,058
9536:095,201,076,240,015,201,124
9542:067,240,015,201,082,240,147
9548:003,076,000,038,162,012,111
9554:208,006,162,008,208,002,164
9560:162,004,173,144,062,041,162
9566:240,141,249,065,138,013,172
9572:249,065,141,249,065,076,177
9578:160,037,032,109,058,240,238
9584:004,162,006,208,002,162,144
9590:014,160,177,169,064,032,222
9596:089,033,032,224,040,240,014
9602:125,160,128,169,004,032,236
9608:000,037,032,056,061,201,011
9614:000,208,111,192,016,176,077
9620:107,173,144,062,041,012,175
9626:141,249,065,152,010,010,013
9632:010,010,013,249,065,141,136
9638:249,065,173,050,062,201,198
9644:006,240,065,173,249,065,202
9650:141,144,062,173,173,065,168
9656:133,132,173,174,065,133,226
9662:133,160,001,177,132,240,009
9668:017,133,131,136,177,132,154
9674:133,130,177,130,041,003,048
9680:013,144,062,145,130,200,134
9686:165,132,024,105,002,133,007
9692:132,165,133,105,000,133,120
9698:133,165,133,197,159,208,197
9704:216,056,032,078,055,076,233
9710:000,038,056,032,078,055,241
9716:144,010,160,000,173,249,212
9722:065,013,232,065,145,130,132
9726:076,152,033,165,134,141,189
9734:238,065,165,135,141,239,221
9740:065,169,003,141,177,065,120
9746:174,178,065,134,134,172,107
9752:179,065,132,135,152,024,199
9758:105,019,141,236,065,189,017
9764:180,065,141,246,065,169,134
9770:000,236,238,065,208,007,028
9776:204,239,065,208,002,169,167
9782:128,141,241,065,152,024,037
9788:105,005,056,237,179,065,195
9794:168,105,071,062,133,143,060
9800:185,095,062,133,142,056,233
9806:032,078,055,176,005,169,081
9812:000,076,220,038,173,232,055
9818:065,240,108,201,002,240,178
9824:104,173,246,065,056,237,209
9830:234,065,170,232,048,048,131
9836:232,173,235,065,041,012,098
9842:201,008,240,038,176,005,014
9848:130,074,240,032,170,142,148
9854:242,065,173,241,065,172,060
9860:177,065,145,142,200,202,039
9066:200,250,140,243,065,173,193
9872:246,065,056,237,242,065,031
9870:170,160,002,076,167,038,251
9884:174,246,065,173,177,065,032
9890:141,243,065,160,002,177,182
9896:130,140,242,065,172,243,136
9902:065,013,241,065,145,142,077
9908:172,242,065,238,243,065,101
9914:202,240,009,200,204,234,251
9920:065,208,228,032,030,039,026
9926:076,235,038,032,186,039,036
9932:174,234,065,202,202,202,003
9938:236,246,065,176,003,076,244
9944:097,038,169,010,013,241,016
9950:065,172,177,065,174,246,097
9956:065,145,142,200,202,208,166
9962:250,164,135,166,134,200,003
9968:204,236,065,240,005,132,098
9974:135,076,035,038,172,179,113
9980:065,132,135,173,246,065,044
9986:024,109,177,065,141,177,183
9992:065,232,134,134,224,051,080
9998:240,039,189,180,065,024,239
10004:109,177,065,201,040,176,020
10010:028,076,035,038,224,000,171
10016:240,020,173,177,065,024,219
10022:109,246,065,168,136,169,163
10028:000,013,241,065,145,142,138
10034:136,202,208,250,096,169,087
10040:040,056,237,177,065,141,004
10046:246,065,160,005,132,135,037
10052:185,071,062,133,143,185,079
10058:095,062,133,142,172,177,087
10064:065,174,246,065,169,000,031
10070:145,142,200,202,208,250,209
10076:230,135,164,135,192,024,204
10082:208,224,173,238,065,133,115
10088:134,173,239,065,133,135,215
10094:160,000,169,000,153,104,184
10100:188,200,192,120,208,248,248
10106:056,032,078,055,144,047,022
10112:160,002,162,000,173,232,089
10118:065,201,002,208,009,172,023
10124:234,065,177,130,141,234,097
10130:065,200,177,130,157,104,211
10136:188,232,200,204,234,065,251
10142:208,244,169,094,157,104,110
10148:188,174,232,065,189,140,128
10154:062,141,103,188,096,169,161
10160:000,141,103,188,169,094,103
10166:141,104,188,096,169,000,112
10172:141,128,004,160,002,177,032
10178:130,201,010,240,242,173,166
10184:235,065,074,074,074,074,028
10190:141,244,065,162,255,201,250
10196:015,240,226,177,130,201,177
10202:014,208,009,174,244,065,164
10208:240,016,232,142,128,004,218
10214:153,127,004,200,204,234,128
10220:065,240,003,202,208,229,159
10226:173,244,065,240,030,224,194
10232:000,240,026,173,128,004,051
10238:201,000,208,010,169,014,088
10244:153,127,004,200,174,244,138
10250:065,232,169,016,153,127,004
10256:004,200,202,208,249,169,024
10262:000,141,128,004,204,234,221
10268:065,240,012,176,063,177,249
10274:130,201,014,240,008,201,060
10280:021,176,012,200,076,096,109
10286:040,200,177,130,201,021,047
10292:144,042,136,152,200,170,128
10298:202,202,189,128,004,201,216
10304:014,240,011,144,012,201,174
10310:025,208,020,169,016,157,153
10316:128,004,202,016,235,202,095
10322:157,128,004,232,169,017,021
10328:157,128,004,208,003,254,074
10334:128,004,136,140,234,065,033
10340:173,128,004,201,000,208,046
10346:009,169,004,133,131,169,209
10352:127,133,130,096,169,004,003
10358:133,131,169,126,133,130,172
10364:238,234,065,096,032,109,130
10370:058,240,005,162,005,076,164
10376:140,040,162,013,169,064,212
10382:160,011,032,089,033,032,243
10388:224,040,160,128,169,004,105
10394:032,000,037,032,056,061,116
10400:201,000,208,053,192,004,050
10406:144,049,192,037,176,045,041
10412:165,134,141,178,065,173,004
10410:050,062,201,005,240,007,231
10424:152,032,126,033,076,197,032
10430:040,152,166,134,157,180,251
10436:065,032,012,037,165,134,129
10442:205,250,065,144,007,172,021
10448:250,065,136,140,178,065,018
10454:032,050,036,076,152,033,081
10460:169,001,208,002,169,000,001
10466:141,245,065,160,000,140,209
10472:240,002,169,032,032,229,160
10470:058,169,126,032,229,050,142
10484:032,028,033,201,155,240,165
10490:051,201,126,240,036,170,050
10496:041,127,201,032,144,230,015
10502:138,174,245,065,208,008,076
10508:201,048,144,228,201,050,124
10514:176,224,166,085,224,038,163
10520:240,218,153,128,004,032,031
10526:229,058,200,208,209,192,102
10532:000,240,205,032,229,058,032
10538:136,076,244,040,162,001,189
10544:142,240,002,032,229,058,239
10550:169,000,153,128,004,140,136
10556:244,065,173,128,004,096,002
10562:165,135,201,100,240,018,157
10568:230,135,173,179,065,024,110
10574:105,018,197,135,176,006,203
10580:238,179,065,032,188,035,053
10586:096,165,135,201,001,240,160
10592:016,198,135,172,179,065,093
10598:136,196,135,144,006,206,157
10604:179,065,032,188,035,096,191
10610:165,134,201,050,240,035,171
10616:230,134,172,240,065,196,133
10622:134,176,026,238,178,065,175
10628:174,178,065,169,000,024,230
10634:125,180,065,232,201,037,210
10640:144,247,202,202,228,134,021
10646:144,233,032,050,036,096,229
10652:165,134,201,001,240,016,145
10658:198,134,172,178,065,136,021
10664:196,134,144,006,206,178,008
10670:065,032,050,036,096,169,110
10676:064,160,018,162,006,032,110
10682:089,033,032,220,040,169,001
10688:004,141,030,062,169,127,213
10694:141,029,062,032,020,062,032
10700:144,078,056,233,065,048,060
10706:073,240,006,201,002,176,140
10712:067,169,026,141,249,065,165
10718:032,020,062,144,057,056,081
10724:233,064,048,052,240,050,147
10730:201,027,176,046,024,109,049
10736:249,065,201,051,176,038,252
10742:141,249,065,032,020,062,047
10748:176,030,032,221,060,032,035
10754:056,061,201,000,208,020,036
10760:192,000,240,016,192,101,237
10766:176,012,192,083,144,011,120
10772:169,082,141,179,065,076,220
10778:034,042,076,152,033,140,247
10784:179,065,132,135,032,012,075
10790:037,173,249,065,205,250,249
10796:065,144,010,172,250,065,238
10802:136,140,178,065,076,060,193
10808:042,141,178,065,133,134,237
10814:032,182,035,076,152,033,060
10820:173,178,065,197,134,208,255
10826:023,173,179,065,197,135,078
```

```
10832:208,016,169,001,141,178,025
10838:065,133,134,141,179,065,035
10844:133,135,032,182,035,096,193
10850:173,178,065,133,134,173,186
10856:179,065,133,135,096,032,232
10862:020,062,141,013,066,032,188
10868:020,062,141,014,066,032,195
10874:020,062,141,015,066,032,202
10880:020,062,201,040,240,003,182
10886:076,242,057,174,207,042,164
10892:173,013,066,221,207,042,094
10898:240,006,202,208,245,076,099
10904:242,057,173,014,066,221,157
10910:216,042,240,002,208,240,082
10916:173,015,066,221,225,042,138
10922:208,232,134,151,224,008,103
10928:176,011,138,072,169,000,230
10934:072,076,181,056,104,133,036
10940:151,032,020,062,166,151,002
10946:202,138,010,170,189,236,115
10952:042,072,189,235,042,072,084
10958:096,009,065,069,073,076,082
10964:083,083,082,083,065,066,162
10970:088,078,079,071,081,078,181
10976:085,086,083,080,084,071,201
10982:078,082,068,077,069,119,211
10988:061,180,061,063,061,186,080
10994:061,126,061,195,061,085,063
11000:061,003,044,098,044,032,010
11006:190,043,142,016,066,140,083
11012:010,066,032,028,062,201,155
11018:058,208,063,032,020,062,197
11024:032,190,043,142,017,066,250
11030:140,019,066,032,028,062,113
11036:201,041,208,044,032,020,062
11042:062,174,016,066,202,236,022
11048:017,066,144,003,076,242,076
11054:057,172,018,066,136,204,187
11060:019,066,144,003,076,242,090
11066:057,232,200,165,134,141,219
11072:247,065,165,135,141,248,041
11078:065,134,134,132,135,096,254
11084:076,242,057,024,032,078,073
11090:055,144,088,160,000,177,194
11096:130,041,003,201,001,240,192
11102:078,200,177,130,141,250,046
11108:065,162,000,200,177,130,066
11114:157,128,004,232,200,204,007
11120:250,065,208,244,173,029,057
11126:062,072,173,030,062,072,077
11132:169,000,157,128,004,169,239
11138:004,160,128,032,000,037,235
11144:104,141,030,062,104,141,206
11150:029,062,165,134,205,017,242
11156:066,240,004,230,134,024,078
11162:096,173,016,066,133,134,004
11168:165,135,205,019,066,240,222
11174:004,230,135,024,096,056,199
11180:096,173,247,065,133,134,252
11186:173,248,065,133,135,024,188
11192:032,078,055,076,242,057,212
11198:162,000,032,028,062,201,163
11204:065,240,006,201,066,208,214
11210:226,162,026,142,249,065,048
11216:032,020,062,201,065,144,220
11222:214,201,091,176,210,056,138
11228:233,064,024,109,249,065,196
11234:201,051,176,199,141,249,219
11240:065,032,020,062,176,191,010
11246:032,221,060,032,056,061,188
11252:201,000,208,101,192,000,002
11258:240,177,192,101,176,173,029
11264:174,249,065,096,169,001,242
11270:141,052,066,169,000,141,063
11276:053,066,032,253,042,032,234
11282:079,043,176,062,165,217,248
11288:072,165,216,072,165,215,161
11294:072,165,214,072,165,213,163
11300:072,165,212,072,238,052,079
11306:066,208,003,238,053,066,164
11312:032,079,043,008,104,141,199
11318:249,065,104,133,224,104,165
11324:133,225,104,133,226,104,217
11330:133,227,104,133,228,104,227
11336:133,229,032,151,061,173,083
11342:249,065,072,040,144,194,074
11348:173,247,065,133,134,173,241
11354:248,065,133,135,024,032,215
11360:078,055,096,032,004,044,149
11366:032,253,061,173,053,066,228
11372:172,052,066,032,182,060,160
11378:032,172,061,096,169,011,143
11384:141,022,066,032,063,035,223
11390:169,125,032,229,058,169,140
11396:008,141,020,066,169,000,024
11402:141,021,066,170,160,091,019
11408:169,065,032,087,033,032,052
11414:220,040,173,128,004,208,155
11420:003,076,154,045,141,051,114
11426:066,162,128,160,004,032,202
11432:126,058,169,065,160,079,057
11438:162,000,032,089,033,169,147
11444:004,174,020,066,172,021,125
11450:066,032,134,058,169,004,137
11456:032,218,058,032,144,058,222
11462:162,004,032,199,058,165,050
11468:134,141,017,066,141,238,173
11474:065,165,135,141,019,066,033
11480:141,239,065,169,001,133,196
11486:134,133,135,169,155,032,212
11492:229,058,166,134,189,180,160
11498:065,141,246,065,170,169,066
11504:000,157,000,006,202,169,006
11510:032,157,000,006,202,016,147
11516:250,056,032,078,055,144,099
11522:091,173,232,065,201,001,253
11528:208,035,173,246,065,056,023
11534:237,234,065,170,232,048,232
11540:020,232,173,235,065,041,018
11546:012,201,008,240,010,176,161
11552:039,138,074,240,004,170,185
11558:076,072,045,162,000,240,121
11564:027,032,186,039,174,234,224
11570:065,202,202,202,236,246,179
11576:065,144,207,174,246,065,189
11582:169,042,157,255,005,202,124
11588:208,250,240,022,160,002,182
11594:177,130,032,204,034,157,040
11600:000,006,232,200,236,246,232
11606:065,240,005,204,234,065,131
11612:208,236,162,000,173,255,102
11618:002,208,251,165,017,240,213
11624:049,189,000,006,240,006,082
11630:032,229,058,232,208,245,090
11636:165,134,205,017,066,240,175
11642:005,230,134,076,230,044,073
11648:165,135,205,019,066,240,190
11654:019,230,135,169,001,133,053
11660:134,169,155,032,229,058,149
11666:173,029,066,048,003,076,029
11672:230,044,169,155,032,229,243
11678:058,169,004,032,218,058,185
11684:032,203,058,173,238,065,165
11690:133,134,173,239,065,133,023
11696:135,173,051,066,201,069,103
11702:208,014,169,158,133,203,043
11708:169,065,133,204,032,110,133
11714:033,032,028,033,032,181,021
11720:035,032,026,035,032,003,107
11726:038,032,152,033,076,182,207
11732:035,169,000,141,254,065,108
11738:141,253,065,032,109,058,108
11744:240,003,238,253,065,165,164
11750:134,141,255,065,165,135,101
11756:141,000,066,076,015,046,068
11762:076,152,033,169,001,141,046
11768:253,065,141,254,065,032,034
11774:109,058,208,003,206,253,067
11780:065,165,134,141,255,065,061
11786:165,135,141,000,066,032,037
11792:076,046,173,238,065,141,243
11798:003,066,173,239,065,141,197
11804:004,066,032,088,046,174,182
11810:255,065,202,236,003,066,093
11816:176,021,174,000,066,202,167
11822:236,004,066,176,012,169,197
11828:064,160,203,162,004,032,165
11834:089,033,032,109,048,173,030
11840:001,066,133,134,173,002,061
11846:066,133,135,076,152,033,153
11852:169,065,160,042,162,004,166
11858:032,089,033,076,097,046,199
11864:169,065,160,002,162,004,138
11870:032,089,033,032,003,038,065
11876:032,028,033,174,158,046,059
11882:221,158,046,240,006,202,211
11888:208,248,076,097,046,202,221
11894:138,010,170,169,046,072,211
11900:169,096,072,189,167,046,095
11906:072,189,166,046,072,096,003
11912:104,104,165,134,141,001,017
11918:066,165,135,141,002,066,205
11924:096,162,253,154,032,152,229
11930:033,076,113,032,007,008,167
11936:028,029,030,031,155,027,204
11942:067,042,090,041,065,041,000
11948:155,041,113,041,135,046,191
11954:148,046,173,007,066,201,051
11960:051,176,091,173,008,066,237
11966:201,101,176,084,173,005,162
11972:066,133,134,173,006,066,006
11978:133,135,056,032,078,055,179
11984:144,069,160,002,173,232,220
11990:065,201,002,208,009,172,103
11996:234,065,177,130,141,234,177
12002:065,200,162,000,177,130,192
12008:157,000,006,232,200,204,007
12014:234,065,208,244,169,000,134
12020:157,000,006,142,234,065,080
12026:032,057,047,173,254,065,110
12032:208,003,032,039,047,173,246
12038:007,066,133,134,173,008,015
12044:066,133,135,024,032,078,224
12050:055,032,174,055,096,173,091
12056:007,066,133,134,173,008,033
12062:066,133,135,024,032,078,242
12068:055,144,239,032,180,053,227
12074:024,032,078,055,169,000,144
12080:160,145,132,200,145,132,202
12086:076,022,047,173,253,065,178
12092:240,001,096,173,232,065,099
12098:201,002,240,001,096,173,011
12104:007,066,056,237,005,066,253
12110:141,011,066,173,008,066,031
12116:056,237,006,066,141,012,090
12122:066,162,000,134,152,189,025
12128:000,006,032,204,034,157,017
12134:104,188,232,236,234,065,137
12140:208,241,169,000,157,104,219
12146:188,169,104,141,029,062,039
12152:169,188,141,030,062,169,111
12158:000,133,203,169,006,133,002
12164:204,032,028,062,032,098,076
12170:048,032,020,062,201,000,245
12176:208,003,076,067,048,201,235
12182:064,208,003,076,046,048,083
12188:144,234,201,067,176,230,184
12194:162,000,201,066,208,002,033
12200:162,026,134,151,032,020,181
12206:062,201,065,144,099,201,178
12212:091,176,095,056,233,064,127
12218:024,101,151,201,051,176,122
12224:085,024,109,011,066,162,137
12230:065,201,027,144,005,162,034
12236:066,056,233,026,024,105,202
12242:064,133,151,138,032,098,058
12248:048,165,151,032,098,048,246
12254:032,020,062,176,051,032,083
12260:221,060,032,056,061,201,091
12266:000,208,041,192,000,240,147
12272:037,192,101,176,033,152,163
12278:024,109,012,066,168,169,026
12284:000,032,182,060,032,190,236
12290:060,162,000,189,031,066,254
12296:240,006,032,098,048,232,152
12302:208,245,032,028,062,076,153
12308:142,047,162,000,189,104,152
12314:188,240,009,032,176,034,193
12320:157,000,006,232,208,242,109
12326:169,000,157,000,006,076,190
12332:092,048,032,098,048,032,138
12338:020,062,032,098,048,032,086
12344:020,062,032,098,048,032,092
12350:020,062,076,136,047,164,055
12356:152,140,234,065,169,000,060
12362:145,203,162,000,189,000,005
12368:006,240,009,032,176,034,065
12374:157,000,006,232,208,242,163
12380:169,128,141,224,188,096,014
12386:164,152,192,120,240,004,202
12392:145,203,230,152,096,173,079
12398:003,066,056,237,255,065,024
12404:024,109,238,065,141,009,190
12410:066,173,004,066,056,237,212
12416:000,066,024,109,239,065,119
12422:141,010,066,173,000,066,078
12428:205,239,065,176,003,076,136
12434:048,049,173,255,065,205,173
12440:238,065,144,074,173,255,077
12446:065,141,005,066,173,000,096
12452:066,141,006,066,173,238,086
12458:065,141,007,066,173,239,093
12464:065,141,008,066,032,180,156
12470:046,173,005,066,205,003,168
12476:066,240,008,238,005,066,043
12482:238,007,066,208,237,173,099
12488:006,066,205,004,066,240,019
12494:020,238,006,066,238,008,014
12500:066,173,255,065,141,005,149
12506:066,173,238,065,141,007,140
12512:066,208,209,076,201,049,009
12518:173,003,066,141,005,066,172
12524:173,009,066,141,007,066,186
12530:173,000,066,141,006,066,182
12536:173,239,065,141,008,066,172
12542:032,180,046,173,005,066,244
12548:205,255,065,240,008,206,215
12554:005,066,206,007,066,208,056
12560:237,173,006,066,205,004,195
12566:066,240,202,238,006,066,072
12572:238,008,066,173,003,066,070
12578:141,005,066,173,009,066,238
12584:141,007,066,208,209,076,235
12590:201,049,173,255,065,205,226
12596:238,065,144,074,173,255,233
12602:065,141,005,066,173,004,000
12608:066,141,006,066,173,238,242
12614:065,141,007,066,173,010,020
12620:066,141,008,066,032,180,057
```

```
12626:046,173,005,066,205,003,068
12632:066,240,008,238,005,066,199
12638:238,007,066,208,237,173,255
12644:006,066,205,000,066,240,171
12650:020,206,006,066,206,008,106
12656:066,173,255,065,141,005,049
12662:066,173,238,065,141,007,040
12668:066,208,209,076,201,049,165
12674:173,003,066,141,005,066,072
12680:173,009,066,141,007,066,086
12686:173,004,066,141,006,066,086
12692:173,010,066,141,008,066,100
12698:032,180,046,173,005,066,144
12704:205,255,065,240,008,206,115
12710:005,066,206,007,066,208,212
12716:237,173,006,066,205,000,091
12722:066,240,020,206,006,066,014
12728:206,008,066,173,003,066,194
12734:141,005,066,173,009,066,138
12740:141,007,066,208,209,076,135
12746:150,051,032,229,058,044,254
12752:029,066,016,005,104,104,020
12758:076,137,050,096,169,064,038
12764:160,043,162,000,032,089,194
12770:033,032,220,040,208,008,255
12776:169,000,141,022,066,076,194
12782:152,033,032,063,035,173,214
12788:244,065,162,128,160,004,239
12794:032,126,058,169,001,162,030
12800:008,160,000,032,134,058,136
12806:032,144,058,048,126,162,064
12812:001,032,199,058,169,254,213
12818:032,204,049,169,254,032,246
12824:204,049,165,136,032,204,046
12830:049,165,137,032,204,049,154
12836:160,050,185,180,065,032,196
12842:204,049,136,208,247,173,035
12848:173,065,133,132,173,174,130
12854:065,133,133,160,001,177,211
12860:132,240,022,165,132,032,015
12866:204,049,165,133,032,204,085
12872:049,136,177,132,032,204,034
12878:049,200,177,132,032,204,104
12884:049,165,132,024,105,002,049
12890:133,132,165,133,105,000,246
12896:133,133,165,133,197,159,248
12902:208,209,169,255,032,204,155
12908:049,165,158,133,132,165,142
12914:159,133,133,160,000,177,108
12920:132,032,204,049,200,208,177
12926:248,230,133,165,133,197,208
12932:137,144,240,240,238,173,024
12938:029,066,072,169,001,032,251
12944:218,058,032,203,058,032,233
12950:026,035,104,141,029,066,039
12956:032,072,059,032,003,038,136
12962:032,182,035,096,032,017,044
12968:059,044,029,066,016,005,131
12974:104,104,076,075,051,096,168
12980:169,064,160,067,162,000,034
12986:032,089,033,032,220,040,120
12992:208,008,169,000,141,022,228
12998:066,076,152,033,032,063,108
13004:035,173,244,065,162,128,243
13010:160,004,032,126,058,169,247
13016:001,162,004,160,000,032,063
13022:134,050,032,144,058,048,184
13028:102,162,001,032,195,058,010
13034:032,166,050,201,254,208,121
13040:126,032,166,050,201,254,045
13046:208,119,032,141,035,032,045
13052:166,050,133,136,032,166,167
13058:050,133,137,160,050,032,052
13064:166,050,153,180,065,136,246
13070:208,247,032,166,050,201,150
13076:255,240,024,133,132,032,068
13082:166,050,133,133,032,166,194
13088:050,160,000,145,132,032,039
13094:166,050,160,001,145,132,180
13100:076,016,051,165,158,133,131
13106:132,165,159,133,133,160,164
13112:000,032,166,050,145,132,069
13118:200,208,248,230,133,165,222
13124:133,197,137,144,240,240,135
13130:238,173,029,066,072,169,053
13136:001,032,218,058,032,203,112
13142:058,044,029,066,016,003,046
13148:032,108,035,032,026,035,104
13154:104,141,029,066,032,072,030
13160:059,032,003,038,076,182,238
13166:035,169,001,032,218,058,111
13172:032,203,058,032,026,035,246
13178:169,065,160,136,162,002,048
13184:032,089,033,032,182,035,019
13190:032,003,038,096,173,029,249
13196:066,201,128,096,173,143,179
13202:062,208,001,096,169,065,235
13208:160,119,162,000,032,089,202
13214:033,165,134,141,238,065,166
13220:165,135,141,239,065,169,054
13226:001,133,134,133,135,173,111
13232:173,065,133,132,173,174,002
13238:065,133,133,160,001,177,083
13244:132,240,053,133,131,136,245
13250:177,132,133,130,177,130,049
13256:041,003,201,002,208,038,181
13262:056,032,078,055,162,000,077
13268:172,234,065,177,130,141,107
13274:234,065,200,177,130,157,157
13280:000,006,232,200,204,234,076
13286:065,208,244,169,000,157,049
13292:000,006,142,234,065,032,203
13298:174,055,165,132,024,105,129
13304:002,133,132,144,002,230,123
13310:133,230,135,165,135,201,229
13316:101,208,178,169,001,133,026
13322:135,230,134,165,134,201,241
13328:051,208,166,173,238,065,149
13334:133,134,173,239,065,133,131
13340:135,056,032,078,055,076,204
13346:152,033,032,180,053,024,252
13352:032,078,055,169,000,168,030
13358:145,132,200,145,132,032,064
13364:144,051,096,169,064,160,224
13370:024,162,008,032,089,033,150
13376:032,109,058,240,008,173,172
13382:143,062,073,255,141,143,119
13388:062,173,143,062,240,006,250
13394:169,078,032,229,058,096,232
13400:169,070,032,229,058,032,166
13406:229,058,096,238,054,062,063
13412:032,205,033,206,054,062,180
13418:173,000,006,240,078,201,036
13424:029,240,039,174,198,032,056
13430:221,198,032,240,008,202,251
13436:208,248,169,001,076,156,214
13442:052,173,234,065,201,037,124
13448:176,051,160,000,169,006,186
13454:032,000,037,032,028,062,077
13460:208,232,169,000,240,002,231
13466:169,002,141,232,065,024,019
13472:032,078,055,176,009,173,171
13478:144,062,141,235,065,076,121
13484:183,052,160,000,177,130,106
13490:041,252,141,235,065,032,176
13496:174,055,032,144,051,096,224
13502:174,244,065,202,202,202,255
13508:202,189,128,004,201,037,189
13514:208,118,232,189,128,004,057
13520:141,249,065,232,189,128,188
13526:004,056,233,016,133,152,040
13532:232,189,128,004,056,233,038
13538:016,166,152,240,006,024,062
13544:105,010,202,208,250,133,116
13550:151,201,098,176,079,173,092
13556:249,065,201,013,240,073,061
13562:162,000,160,000,189,128,121
13568:004,201,037,240,008,232,210
13574:201,014,240,244,200,208,089
13580:241,136,140,249,065,165,240
13586:151,056,237,249,065,133,141
13592:151,162,001,160,001,189,176
13598:128,004,232,201,014,240,081
13604:248,201,037,240,006,153,153
13610:128,004,200,208,238,169,221
13616:016,166,151,153,128,004,154
13622:200,202,208,249,169,000,058
13628:153,128,004,140,244,065,026
13634:096,198,151,162,000,160,065
13640:000,189,128,004,232,201,058
13646:014,240,248,201,037,240,034
13652:006,153,104,188,200,208,175
13658:238,169,000,153,104,188,174
13664:169,014,141,128,004,166,206
13670:151,169,016,157,128,004,215
13676:202,208,250,162,000,164,070
13682:151,200,189,104,188,153,075
13688:128,004,240,004,232,200,160
13694:208,244,140,244,065,096,099
13700:032,140,033,169,000,133,127
13706:085,133,084,173,175,065,085
13712:056,229,136,168,173,176,058
13718:065,229,137,032,162,053,060
13724:169,000,141,022,066,096,138
13730:032,182,060,032,190,060,206
13736:169,066,133,204,169,031,172
13742:133,203,032,110,033,096,013
13748:160,001,177,132,240,231,097
13754:169,000,145,132,136,145,145
13760:132,177,130,041,003,201,108
13766:002,208,009,200,177,130,156
13772:168,177,130,076,213,053,253
13778:200,177,130,133,203,024,053
13784:101,130,141,247,053,165,029
13790:130,141,250,053,165,131,068
13796:141,251,053,105,000,141,151
13802:248,053,165,137,056,237,106
13808:248,053,170,232,160,000,079
13814:185,255,255,153,255,255,068
13820:200,208,247,238,248,053,166
13826:238,251,053,202,208,238,168
13832:165,136,056,229,203,133,162
13838:136,165,137,233,000,133,050
13844:137,173,173,065,133,156,089
13850:173,174,065,133,157,160,120
13856:001,177,156,240,032,056,182
13862:136,177,156,229,130,133,231
13868:151,200,177,156,229,131,064
13874:005,151,144,015,136,177,166
13880:156,056,229,203,145,156,233
13886:200,177,156,233,000,145,205
13892:156,200,240,003,200,208,051
13898:214,230,157,200,165,157,173
13904:197,159,208,205,096,169,090
13910:062,160,247,162,002,032,239
13916:089,033,032,028,033,041,092
13922:095,201,089,208,003,108,034
13928:010,000,076,152,033,173,036
13934:247,065,133,134,173,248,086
13940:065,133,135,024,032,078,071
13946:055,173,249,065,141,232,013
13952:065,173,251,065,141,235,034
13958:065,173,250,065,141,234,038
13964:065,076,242,057,072,165,049
13970:134,141,247,065,165,135,009
13976:141,248,065,173,232,065,052
13982:141,249,065,173,235,065,062
13988:141,251,065,173,234,065,069
13994:141,250,065,104,233,065,004
14000:048,187,240,006,201,002,092
14006:176,181,169,026,133,134,233
14012:032,020,062,233,064,048,135
14018:170,240,168,201,027,176,152
14024:164,024,101,134,201,051,107
14030:176,157,133,134,032,020,090
14036:062,176,150,032,221,060,145
14042:032,056,061,201,000,208,008
14048:140,192,000,240,136,192,100
14054:101,176,132,132,135,056,194
14060:032,078,055,144,007,173,213
14066:232,065,201,001,208,003,184
14072:076,109,054,160,002,162,043
14078:000,177,130,201,010,240,244
14084:243,177,130,157,128,004,075
14090:200,232,204,234,065,208,129
14096:244,169,000,157,128,004,206
14102:173,029,062,072,173,030,049
14108:062,072,160,128,169,004,111
14114:032,000,037,104,141,030,122
14120:062,104,141,029,062,173,099
14126:247,065,133,134,173,248,022
14132:065,133,135,024,032,078,007
14138:055,173,249,065,141,232,205
14144:065,173,251,065,141,235,226
14150:065,173,250,065,141,234,230
14156:065,096,008,166,134,202,235
14162:134,132,169,100,133,133,115
14168:024,169,000,162,008,106,045
14174:102,132,144,003,024,101,088
14180:133,202,016,245,133,133,194
14186:166,135,202,138,024,101,104
14192:132,133,132,165,133,105,144
14198:000,133,133,006,132,038,0^8
14204:133,165,133,109,174,065,135
14210:133,133,160,001,177,132,098
14216:208,003,040,024,096,170,165
14222:136,177,132,133,130,134,216
14228:131,040,144,020,177,130,022
14234:041,003,141,232,065,177,045
14240:130,041,252,141,235,065,000
14246:200,177,130,141,234,065,089
14252:056,096,032,180,053,173,250
14258:232,065,201,002,240,050,200
14264:238,234,065,238,234,065,234
14270:160,000,165,136,145,132,160
14276:200,165,137,145,132,136,087
14282:173,232,065,013,235,065,217
14288:145,136,200,173,234,065,137
14294:145,136,200,162,000,189,022
14300:000,006,145,136,200,232,171
14306:204,234,065,208,244,076,233
14312:061,056,032,128,056,238,035
14318:244,065,238,244,065,056,126
14324:173,244,065,109,234,065,110
14330:141,234,065,172,244,065,147
14336:173,234,065,145,136,162,147
14342:000,200,189,000,006,145,034
14348:136,200,232,204,234,065,059
14354:208,244,160,000,165,136,163
14360:145,132,200,165,137,145,180
14366:132,136,173,232,065,013,013
14372:235,065,145,136,200,173,222
14378:244,065,145,136,200,162,226
14384:002,189,126,004,145,136,138
14390:200,232,236,244,065,208,215
14396:244,165,136,024,109,234,204
14402:065,144,006,165,137,201,016
14408:186,240,015,165,136,024,070
14414:109,234,065,133,136,165,152
14420:137,105,000,133,137,096,180
```

```
14426:169,255,141,252,002,169,054
14432:000,168,145,132,200,145,118
14438:132,169,064,160,228,162,249
14444:000,032,089,033,165,134,049
14450:141,178,065,165,135,141,171
14456:179,065,162,253,154,076,241
14462:113,032,186,142,252,065,148
14468:162,000,160,000,189,000,131
14474:006,032,204,034,201,040,143
14480:208,001,200,201,041,208,235
14486:001,136,157,000,006,232,170
14492:236,234,065,208,231,192,042
14498:000,240,003,076,242,057,012
14504:169,000,072,169,000,141,207
14510:029,062,169,006,141,030,099
14516:062,032,020,062,144,082,070
14522:201,045,240,078,201,043,226
14528:240,074,201,046,240,070,039
14534:201,080,240,037,201,040,229
14540:240,021,201,065,240,011,214
14546:201,066,240,007,201,064,221
14552:240,015,076,242,057,032,110
14558:144,054,076,015,057,169,225
14564:001,072,076,181,056,032,134
14570:109,042,076,015,057,032,053
14576:020,062,201,073,240,003,071
14582:076,242,057,169,006,160,188
14588:057,032,177,060,032,020,118
14594:062,076,015,057,064,003,023
14600:020,021,146,101,032,221,037
14606:060,032,028,062,240,119,043
14612:162,002,201,043,240,055,211
14618:232,201,045,240,050,232,002
14624:201,042,240,045,232,201,225
14630:047,240,040,232,201,094,124
14636:240,035,201,041,240,003,036
14642:076,242,057,104,240,022,023
14648:201,001,240,007,072,032,097
14654:194,057,076,053,057,238,225
14660:029,062,208,003,238,030,126
14666:062,076,015,057,076,186,034
14672:042,134,207,104,072,168,039
14678:185,119,062,221,119,062,086
14684:144,016,032,194,057,166,189
14690:207,104,072,168,185,119,185
14696:062,221,119,062,176,240,216
14702:165,217,072,165,216,072,249
14708:165,215,072,165,214,072,251
14714:165,213,072,165,212,072,253
14720:165,207,072,076,181,056,117
14726:240,106,076,172,061,104,125
14732:072,240,006,032,194,057,229
14738:076,139,057,104,032,190,232
14744:060,160,000,185,031,066,142
14750:240,009,032,176,034,153,034
14756:128,004,200,208,242,153,075
14762:128,004,140,244,065,162,145
14768:000,189,000,006,240,009,108
14774:032,176,034,157,000,006,075
14780:232,208,242,076,190,052,164
14786:104,133,203,104,133,204,051
14792:104,133,145,104,133,224,019
14798:104,133,225,104,133,226,107
14804:104,133,227,104,133,228,117
14810:104,133,229,165,145,010,236
14816:168,165,204,072,165,203,177
14822:072,185,127,062,072,185,165
14828:126,062,072,165,212,096,201
14834:174,252,065,154,169,007,039
14840:141,244,065,160,000,185,019
14846:004,064,153,128,004,200,039
14852:192,007,208,245,169,000,057
14858:153,128,004,076,175,057,091
14864:169,125,032,229,058,032,149
14870:063,035,169,005,162,104,048
14876:160,058,032,126,058,169,119
14882:007,162,006,160,000,032,145
14888:134,058,032,218,058,032,060
14894:144,058,048,019,162,007,228
14900:032,195,058,032,017,059,189
14906:044,029,066,048,006,032,027
14912:229,058,076,055,058,032,060
14918:203,058,169,007,032,218,245
14924:058,169,158,133,203,169,198
14930:065,133,204,032,110,033,147
14936:032,028,033,032,026,035,018
14942:032,003,038,032,152,033,128
14948:032,102,035,096,068,058,059
14954:042,046,042,169,008,141,042
14960:031,208,173,031,208,201,196
14966:003,096,010,010,010,010,001
14972:170,096,141,023,066,134,242
14978:149,132,150,096,141,024,054
14984:066,142,026,066,140,025,089
14990:066,096,173,024,066,032,087
14996:120,058,165,149,157,068,097
15002:003,165,150,157,069,003,189
15008:173,023,066,157,072,003,142
15014:169,000,157,073,003,173,229
15020:026,066,157,074,003,173,159
15026:025,066,157,075,003,169,161
15032:003,157,066,003,032,086,019
15038:228,140,029,066,096,142,123
15044:027,066,096,142,028,066,109
15050:096,162,000,142,027,066,183
15056:142,028,066,142,024,066,164
15062:142,029,066,096,032,120,187
15068:058,169,012,157,066,003,173
15074:076,188,058,141,120,059,100
15080:140,121,059,142,122,059,107
15086:173,028,066,032,120,058,203
15092:169,000,157,072,003,157,034
15098:073,003,169,011,157,066,217
15104:003,173,120,059,032,188,063
15110:058,172,121,059,174,122,200
15116:059,173,120,059,096,140,147
15122:121,059,142,122,059,173,182
15128:027,066,032,120,058,169,240
15134:000,157,072,003,157,073,236
15140:003,169,007,157,066,003,185
15146:032,188,058,172,121,059,160
15152:174,122,059,009,000,096,252
15158:162,007,142,123,059,138,173
15164:032,218,058,174,123,059,212
15170:202,208,243,076,203,058,032
15176:173,029,066,133,151,048,160
15182:010,169,064,160,091,162,222
15188:005,032,089,033,096,201,028
15194:128,208,010,169,064,160,061
15200:119,162,007,032,089,033,026
15206:096,169,064,160,107,162,092
15212:002,032,089,033,164,151,067
15218:169,000,032,162,053,096,114
15224:058,032,002,000,043,173,172
15230:252,002,201,255,208,003,023
15236:169,000,096,173,252,002,056
15242:201,255,240,249,141,124,068
15248:059,169,255,141,252,002,254
15254:133,017,032,218,059,173,014
15260:124,059,201,192,176,016,156
15266:041,063,201,060,208,024,247
15272:173,124,059,041,064,240,101
15278:006,141,190,002,169,000,170
15284:096,173,190,002,073,064,010
15290:141,190,002,169,000,096,016
15296:174,124,059,189,241,059,014
15302:044,190,002,080,010,201,213
15308:097,144,006,201,123,176,183
15314:002,041,223,201,128,240,021
15320:217,096,072,169,100,141,243
15326:000,210,162,175,142,001,144
15332:210,160,128,136,208,253,043
15338:202,224,159,208,243,104,094
15344:096,108,106,059,128,128,097
15350:107,043,042,111,128,112,021
15356:117,155,105,045,061,118,085
15362:128,099,128,128,098,120,191
15368:122,052,128,051,054,027,186
15374:053,050,049,044,032,046,032
15380:110,128,109,047,128,114,144
15386:128,101,121,127,116,119,226
15392:113,057,128,048,055,126,047
15398:056,060,062,102,104,100,010
15404:128,130,103,115,097,076,181
15410:074,050,128,128,075,092,093
15416:094,079,128,080,085,155,165
15422:073,095,124,086,128,067,123
15428:128,128,066,088,090,036,092
15434:128,035,038,027,037,034,117
15440:033,091,032,093,078,128,023
15446:077,063,128,082,128,069,121
15452:089,159,084,087,081,040,120
15458:128,041,039,156,064,125,139
15464:157,070,072,068,128,131,218
15470:071,083,065,012,010,123,218
15476:128,128,011,030,031,015,203
15482:128,016,021,155,009,028,223
15488:029,022,128,003,128,128,054
15494:002,024,026,128,128,133,063
15500:128,027,128,253,128,000,036
15506:032,096,014,128,013,128,045
15512:128,010,128,005,025,158,102
15518:020,023,017,128,128,128,090
15524:128,254,128,125,255,006,036
15530:008,004,128,132,007,019,212
15536:001,170,032,137,221,096,065
15542:132,212,133,213,032,170,050
15548:217,096,032,230,216,160,115
15554:000,177,243,048,006,153,053
15560:031,066,200,208,246,041,224
15566:127,153,031,066,169,000,240
15572:200,153,031,066,169,031,094
15578:160,066,096,173,029,062,036
15584:133,243,173,030,062,133,230
15590:244,169,000,133,242,032,026
15596:006,061,032,000,216,032,071
15602:036,061,024,173,029,062,115
15608:101,242,141,029,062,173,228
15614:030,062,105,000,141,030,110
15620:062,096,160,000,140,030,236
15626:066,177,243,201,032,144,105
15632:001,096,169,255,141,030,196
15638:066,177,243,240,008,032,020
15644:204,034,145,243,200,208,038
15650:244,096,173,030,066,016,147
15656:250,160,000,177,243,240,086
15662:244,032,176,034,145,243,152
15668:200,206,244,096,032,210,018
15674:217,164,212,165,213,096,101
15680:165,212,041,127,056,233,130
15686:062,170,016,002,162,000,226
15692:169,000,149,212,232,224,038
15698:006,208,249,096,165,212,250
15704:141,019,062,041,127,133,099
15710:212,162,210,160,061,032,163
15716:152,221,032,151,061,032,237
15722:064,061,173,019,062,016,245
15728:006,165,212,009,128,133,253
15734:212,096,165,212,041,127,203
15740:133,212,096,160,000,165,122
15746:212,072,240,002,160,001,049
15752:169,000,032,182,060,104,171
15758:016,006,165,212,073,128,230
15764:133,212,096,032,102,218,173
15770:176,037,096,032,253,061,041
15776:032,096,218,176,028,096,038
15782:032,219,218,176,022,096,161
15788:032,253,061,032,040,219,041
15794:176,013,096,032,192,221,140
15800:176,007,096,032,205,222,154
15806:176,001,096,076,242,057,070
15812:032,182,221,169,210,160,146
15818:061,032,177,060,032,216,012
15824:061,096,063,080,000,000,252
15830:000,000,162,013,160,062,099
15836:032,167,221,032,243,061,208
15842:032,187,061,162,013,160,073
15848:062,032,152,221,032,219,182
15854:218,032,181,061,096,162,220
15860:005,181,224,149,212,202,193
15866:016,249,096,162,005,181,191
15872:224,072,181,212,149,224,038
15878:104,149,212,202,016,243,164
15884:096,000,000,000,000,000,108
15890:000,000,238,029,062,208,043
15896:003,238,030,062,173,128,146
15902:004,201,058,240,011,176,208
15908:008,201,032,056,233,048,102
15914:056,233,208,096,169,058,094
15920:096,000,002,040,002,000,188
15926:000,000,014,050,150,086,098
15932:194,114,046,068,050,056,076
15938:004,008,198,118,010,188,080
15944:188,188,188,188,189,189,178
15950:189,189,189,189,189,190,189
15956:190,190,190,190,190,191,201
15962:191,191,191,191,191,064,085
15968:104,144,184,224,008,048,040
15974:088,128,168,208,248,032,206
15980:072,112,152,192,232,016,116
15986:056,096,136,176,216,000,026
15992:001,002,002,003,003,004,135
15998:148,057,148,057,150,061,235
16004:156,061,165,061,171,061,039
16010:215,061,046,052,038,000,038
16016:044,000,000,033,033,033,031
16022:034,033,035,033,036,033,098
16028:037,033,038,033,039,033,113
16034:040,033,041,033,042,033,128
16040:043,033,044,033,045,033,143
16046:046,033,047,033,048,033,158
16052:049,033,050,033,051,033,173
16058:052,033,053,033,054,033,188
16064:055,033,056,033,057,033,203
16070:058,034,033,034,034,034,169
16076:035,034,036,034,037,034,158
16082:038,034,039,034,040,034,173
16088:041,034,042,034,043,034,188
16094:044,034,045,034,046,034,203
16100:047,034,048,034,049,034,218
16106:050,034,051,034,052,034,233
16112:053,034,054,034,055,034,248
16118:056,069,088,073,084,032,136
16124:116,111,032,068,079,083,229
16130:058,032,065,114,101,032,148
16136:121,111,117,032,115,117,109
16142:114,101,063,032,040,089,197
16148:047,078,041,058,000,083,071
16154:112,101,101,100,067,097,092
16160:108,099,000,125,083,112,047
16166:101,101,100,067,097,108,100
16172:099,032,066,121,032,075,213
16178:046,032,077,097,114,116,020
16184:105,110,032,038,032,067,184
16190:046,032,066,114,097,110,015
16196:110,111,110,000,083,089,059
16202:083,084,069,077,032,210,117
16208:197,211,197,212,032,084,245
16214:114,097,112,112,101,100,210
16220:046,000,125,089,111,117,068
```

```
16226:032,109,117,115,116,032,107
16232:104,097,118,101,032,052,096
16238:056,075,032,116,111,032,020
16244:114,117,110,032,083,112,172
16250:101,101,100,067,097,108,184
16256:099,046,155,155,082,101,254
16262:109,111,118,101,032,097,190
16268:110,121,032,099,097,114,201
16274:116,114,105,100,103,101,017
16280:115,044,032,111,114,032,088
16286:111,110,032,080,076,047,110
16292:008,069,155,104,111,108,031
16298:100,032,100,111,119,110,230
16304:032,207,208,212,201,207,219
16310:206,032,097,115,032,121,017
16316:111,117,032,114,101,045,196
16322:098,111,111,116,046,155,063
16328:155,080,114,101,115,115,112
16334:032,210,197,212,213,210,000
16340:206,032,116,111,032,114,055
16346:101,045,098,111,111,116,032
16352:058,000,067,076,069,065,047
16358:082,032,083,072,069,069,125
16364:084,058,032,065,114,101,178
16370:032,121,111,117,032,115,002
16376:117,114,101,063,032,040,203
16382:089,047,078,041,058,000,055
16388:010,101,114,114,111,114,056
16394:010,087,105,100,116,104,020
16400:058,000,071,111,116,111,227
16406:058,000,082,101,099,097,203
16412:108,099,117,108,097,116,161
16418:105,111,110,032,105,115,100
16424:032,079,000,083,097,118,193
16430:101,032,040,068,101,118,250
16436:105,099,101,058,070,105,078
16442:108,101,110,097,109,101,172
16448:041,062,000,076,111,097,195
16454:100,032,040,068,101,118,017
16460:105,099,101,058,070,105,102
16466:108,101,110,097,109,101,196
16472:041,062,000,079,075,046,135
16478:032,032,078,111,032,101,224
16484:114,114,111,114,115,046,202
16490:000,073,047,079,032,069,150
16496:114,114,111,114,032,035,120
16502:000,032,066,114,101,097,016
16508:107,032,107,101,121,032,112
16514:097,098,111,114,116,033,187
16520:000,070,111,114,109,097,125
16526:116,058,032,204,101,102,243
16532:116,044,032,195,101,110,234
16538:116,101,114,044,032,111,160
16544:114,032,210,105,103,104,060
16550:116,032,074,117,115,116,224
16556:105,102,121,063,000,078,129
16562:117,109,098,101,114,032,237
16568:111,102,032,100,101,099,217
16574:105,109,097,108,032,112,241
16580:100,097,099,101,115,063,011
16586:000,080,114,111,099,101,195
16592:115,115,105,110,103,032,020
16598:100,097,116,097,032,116,004
16604:114,097,110,115,102,101,091
16610:114,000,076,111,116,032,165
16616:101,110,111,117,103,104,110
16622:032,114,111,111,109,032,235
16628:116,111,032,101,110,116,062
16634:101,114,032,100,097,116,042
16640:097,000,077,111,118,101,248
16646:032,099,117,114,115,111,082
16652:114,032,116,111,032,116,021
16658:111,112,032,108,101,102,072
16664:116,032,111,102,032,110,015
16670:101,119,032,112,111,115,108
16676:105,116,105,111,110,000,071
16682:077,111,118,101,032,099,068
16688:117,114,115,111,114,032,139
16694:116,111,032,098,111,116,126
16700:116,111,109,032,114,105,135
16706:103,104,116,032,111,102,122
16712:032,098,108,111,099,107,115
16718:000,080,114,105,110,116,091
16724:105,110,103,046,046,046,028
16730:000,080,114,105,110,116,103
16736:032,116,111,032,040,068,239
16742:101,118,105,099,101,058,172
16748:070,105,108,101,110,097,187
16754:109,101,041,062,000,082,253
16760:101,099,097,108,099,117,229
16766:108,097,116,105,110,103,253
16772:046,046,046,000,078,111,203
16778:116,032,097,032,083,112,098
16784:101,101,100,067,097,108,206
16790:099,032,102,105,108,101,185
16796:046,000,155,080,114,101,140
16802:115,115,032,210,197,212,019
16808:213,210,206,155,000,000,184
```


IBM Fractal Graphics

Paul W. Carlson

One of the hottest topics in mathematics these days is fractals—*fractional dimensions. Fractals are being used for everything from simulating random plant growth to generating realistic planetary landscapes for science-fiction films and arcade games. This article, adapted from "Apple Fractals" in the September 1985 issue of* COMPUTE!, *introduces the fascinating world of fractals with three programs that work on any IBM PCjr or PC with color/graphics adapter.*

The term *fractal* was coined by Benoit Mandelbrot, a pioneer in their study, to denote curves or surfaces having *fractional dimension*. The concept of fractional dimension can be illustrated as follows: A straight curve (a line) is one-dimensional, having only length. However, if the curve is infinitely long and curves about in such a manner as to completely fill an area of the plane containing it, the curve could be considered two-dimensional. A curve partially filling an area would have a fractional dimension between one and two.

Many types of fractals are *self-similar*, which means that all portions of the fractal resemble each other. Self-similarity occurs whenever the whole is an expansion of some basic building block. In the language of fractals, this basic building block is called the *generator*. The generator in the accompanying programs consists of a number of connected line segments. The curves that the programs plot are the result of starting with the generator and then repeatedly replacing each line segment with the whole generator according to a defined rule. Theoretically, these replacement cycles would continue indefinitely. In practice, the screen resolution limits the number of cycles.

The programs illustrate two types of fractal curves. The curves generated by Program 1 and Program 2 are *self-contacting*, while the curve generated by Program 3 is *self-avoiding*. A self-contacting curve touches itself but does not cross itself. A self-avoiding curve never actually touches itself although it may appear to because of the limited screen resolution.

The Dragon Sweep

Program 1 plots what Mandelbrot refers to as a "dragon sweep." It demonstrates in a step-by-step fashion how a fractal curve is filled.

The generator consists of two line segments of equal length forming a right angle. During each replacement cycle, the generator is substituted for each segment on alternating sides of the segments, that is, to the left of the first segment, to the right of the second segment, and so on. Figure 1 shows the first few cycles of substitution. The program is written in BASIC so the plotting is slow enough to let you observe the development of the curve.

The program prompts you to enter an even number of cycles (for reasons of efficiency and screen resolution, only even numbers of cycles are plotted). When a plot is complete, pressing any key clears the screen and returns you to the prompt. I recommend starting with two cycles, then four, six, etc. It takes fourteen cycles to completely fill in the "dragon," but since this requires almost two hours, you will probably want to quit after about ten cycles. You can see the complete dragon by running Program 2, which always plots the dragon first in less than 30 seconds.

Since it's not at all obvious how the program works, here's a brief explanation. NC is the number of cycles; C is the cycle number; SN is an array of segment numbers indexed by cycle number; L is the segment length; D is the segment direction, numbered clockwise from the positive x direction; and X and Y are the high-resolution screen coordinates.

Lines 100–140	Get number of cycles from user.
Line 150	Computes segment length.
Line 160	Sets starting coordinates.
Line 170	Sets segment numbers for all cycles to the first segment.
Lines 180–220	Find the direction of the segment in the last cycle by rotating the segment in each cycle that will contain the segment in the last cycle.
Lines 230–260	Increase or decrease X or Y by the segment length, depending on the segment direction.
Lines 270–290	Plot the segment and update the current segment number for each cycle.
Lines 300–320	If the segment number for cycle zero is still zero, do the next segment; otherwise, we're done.

Eight Thousand Dragons

Program 2 plots more than 8,000 different dragons. It does this by randomly determining on which side of the first segment the generator will be substituted for all cycles after the first cycle. The generator is always substituted to the left of the first segment in the first cycle to avoid plotting off the screen. Other than the randomization, this program uses the same logic as Program 1. The main part of this program is written in machine language to reduce the time required to plot a completely filled-in dragon from about two hours to less than half a minute.

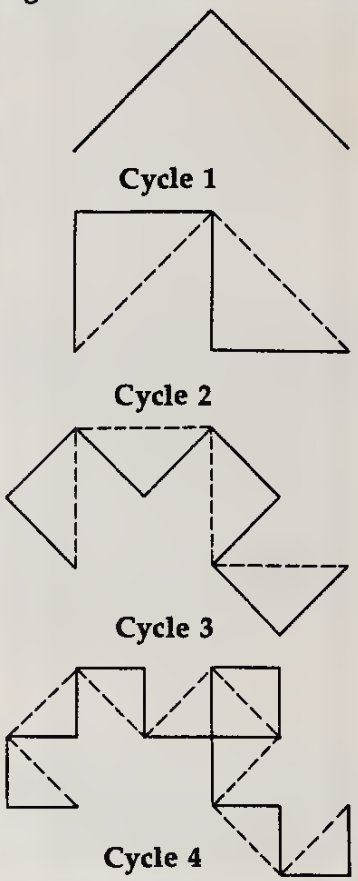
All the dragons are plotted after 14 cycles of substitution. All have exactly the same area, which equals half of the square of the distance between the first and last points plotted. All the dragons begin and end at the same points.

When a plot is complete, press the space bar to plot another dragon, or press the Q key to quit.

Snowflakes

Program 3 plots what Mandelbrot refers to as a "snowflake sweep." The generator, shown in Figure 2, was discovered by Mandelbrot. The segments are numbered zero through six, starting at the right. The program is basically the same as Program 1. The variables NC, C, SN, D, X, and Y represent the same values except that the direction D is numbered counterclockwise from the negative x direction. For each segment, the accompanying table gives the value of RD (relative direction), LN (length factor), and SD (flags indicating which side of the segment the generator is to be placed).

Figure 1: Substitution Cycles, Program 1



Line 20	Reads values of SD and RD. Compute LN values.
Lines 30–50	Compute delta x and delta y factors for each direction.
Lines 60–100	Get number of cycles from user.
Line 120	Sets starting coordinates.
Line 130	Sets the segment numbers for all cycles to the first segment.
Lines 140–170	Find the direction of the segment in the last cycle.

Lines 180–190 Compute the coordinates of the end of the segment, plot the segment, and update the segment numbers for each cycle.

Lines 200–220 Same as lines 300–320 in Program 1.

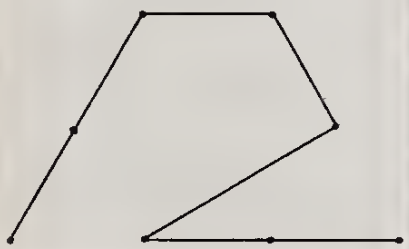
Like Program 1, pressing any key when a plot is complete clears the screen and brings another prompt.

Experiment

I hope these programs encourage you to look further into the fascinating world of fractals. Don't be afraid to experiment with the programs—try modifying the shape of the generator in Program 3, for example. Better yet, design your own generator.

These programs just begin to explore the possibilities of fractal computer graphics. There is another whole class of fractals, those generated by functions of complex variables. And then there are three-dimensional fractals. And then...

Figure 2: Generator, Program 3



Values For Program 3

Segment Number SN	Relative Direction RD	Length Factor LN	Side Flag SD
0	0	1/3	0
1	0	1/3	1
2	7	$\sqrt{1/3}$	1
3	10	1/3	0
4	0	1/3	0
5	2	1/3	0
6	2	1/3	1

Program 1: The Dragon Sweep

```
NH 90 DIM SN(14):KEY DFF
MN 100 CLS:SCREEN 0
PI 110 PRINT"ENTER AN EVEN ND. O
       F CYCLES (2 TO 14)"
00 120 INPUT"           DR ENTER
       A ZERD TD QUIT: ";NC
EH 130 IF NC = 0 THEN KEY DN:END
MO 140 IF NC MDD 2 = 1 DR NC < 2
        DR NC > 14 THEN 100
OL 150 L=128:FDR C=2 TO NC STEP
       2:L=L/2:NEXT
AE 160 X=192:Y=133:CLS:SCREEN 2:
       PSET (X,Y),1
OL 170 FDR C=0 TD NC:SN(C)=0:NEX
       T
KJ 180 D=0:FDR C=1 TD NC:IF SN(C
       -1)=SN(C) THEN D=D-1:GDTD
        200
BC 190 D=D+1
BP 200 IF D=-1 THEN D=7
MI 210 IF D=8 THEN D=0
NC 220 NEXT
OA 230 IF D=0 THEN X=X+L+L:GDTD
       270
EM 240 IF D=2 THEN Y=Y+L:GDTD 27
       0
GA 250 IF D=4 THEN X=X-L-L:GDTD
       270
GM 260 Y=Y-L
PL 270 LINE -(X,Y),1:SN(NC)=SN(N
       C)+1
OP 280 FDR C=NC TD 1 STEP -1:IF
       SN(C)<>2 THEN 300
OH 290 SN(C)=0:SN(C-1)=SN(C-1)+1
       :NEXT
GA 300 IF SN(0)=0 THEN 180
MM 310 IF INKEY$="" THEN 310
AC 320 GDTD 100
```

Program 2: Eight Thousand Dragons

```
PP 100 DEF SEG:CLEAR,&H3FF0:N=&H
       4000
LL 110 READ A$:IF A$="/" THEN 13
       0
LI 120 PDKE N,VAL("&H"+A$):N=N+1
       :GDTD 110
OC 130 N=&H440F:FDR K=1 TD 15:PD
       KE N,0:N=N+1:NEXT
NH 140 PDKE &H4425,0
IF 150 N=&H4000:CALL N:PDKE &H44
       25,1
EA 160 A$=INKEY$:IF A$="" THEN 1
       60
PI 170 IF A$=" " THEN 150
IO 180 IF A$<>"Q" AND A$<>"q" TH
       EN 160
IH 190 SCREEN 0:CLS:KEY ON:END
EI 1000 DATA 1E,0E,1F,88,05,00,C
        D,10,80,3E
ON 1010 DATA 25,44,00,75,08,84,0
        0,CD,1A,89
NF 1020 DATA 16,23,44,E8,31,90,8
        E,02,00,89
DE 1030 DATA 08,00,A1,23,44,33,D
        2,A9,02,00
JE 1040 DATA 74,02,B2,01,A9,04,0
        0,74,02,86
MG 1050 DATA 01,32,D6,D0,EA,D1,D
        8,E2,E8,A3
BJ 1060 DATA 23,44,24,01,88,84,0
        F,44,46,83
FH 1070 DATA FE,0F,75,D3,88,00,0
        6,33,C9,8A
LI 1080 DATA 4F,18,32,FF,CD,10,8
        9,0F,00,33
MJ 1090 DATA F6,C6,84,00,44,00,4
        6,E2,F8,C7
JO 1100 DATA 06,1E,44,60,00,C7,0
        6,20,44,84
HF 1110 DATA 00,88,01,0C,88,0E,1
        E,44,88,16
DC 1120 DATA 20,44,CD,10,C6,06,2
        2,44,00,89
LO 1130 DATA 0E,00,33,FF,8E,01,0
        0,8A,A5,0F
KC 1140 DATA 44,80,FC,00,75,18,F
        E,06,22,44
CL 1150 DATA 8A,85,00,44,3A,84,0
        0,44,75,20
LG 1160 DATA FE,0E,22,44,FE,0E,2
        2,44,E8,16
DO 1170 DATA FE,0E,22,44,8A,85,0
        0,44,3A,84
HK 1180 DATA 00,44,75,08,FE,06,2
        2,44,FE,06
OC 1190 DATA 22,44,80,3E,22,44,F
        F,75,07,C6
HN 1200 DATA 06,22,44,07,E8,0C,B
        0,3E,22,44
NM 1210 DATA 08,75,05,C6,06,22,4
        4,00,47,46
FE 1220 DATA E2,A8,E8,02,E8,9A,8
        0,3E,22,44
CH 1230 DATA 00,75,06,FF,06,1E,4
        4,E8,1E,80
KP 1240 DATA 3E,22,44,02,75,06,F
        F,06,20,44
IG 1250 DATA E8,11,80,3E,22,44,0
        4,75,06,FF
CO 1260 DATA 0E,1E,44,E8,04,FF,0
        E,20,44,88
ID 1270 DATA 01,0C,88,0E,1E,44,8
        8,16,20,44
LL 1280 DATA CD,10,FE,06,0E,44,8
        F,0D,00,8E
KL 1290 DATA 0E,00,89,0E,00,80,8
        C,00,44,02
PI 1300 DATA 75,0D,C6,84,00,44,0
        0,FE,85,00
FM 1310 DATA 44,4F,4E,E2,EC,80,3
        E,00,44,00
OK 1320 DATA 75,02,E8,9C,1F,C8,/
```

Program 3: The Snowflake Sweep

```
OE 10 DIM DX(11),DY(11):KEY DFF
MC 20 FOR N = 0 TD 6:READ SD(N),
      RD(N):LN(N)=1!/3!:NEXT:LN(
      2)=SQR(LN(1))
LC 30 A=0:FOR D=6 TD 11:DX(D)=CD
      S(A):DY(D)=SIN(A)
JN 40 A=A+.52359879#:NEXT
CK 50 FDR D=0 TD 5:DX(D)=-DX(D+6
      ):DY(D)=-DY(D+6):NEXT:X1=5
      34:Y1=147:TL=324
OG 60 CLS:SCREEN 0
AO 70 PRINT"ENTER NUMBER DF CYCL
      ES ( 1 - 4 )"
GK 80 INPUT "          OR ENTER A
      ZERD TD QUIT: ";NC
JA 90 IF NC = 0 THEN END
DA 100 IF NC > 4 THEN 60
OP 110 CLS:SCREEN 2
CL 120 X=534:Y=147:TL=324:PSET (
      X,Y),1
CD 130 FOR C=0 TO NC:SN(C)=0:NEX
       T
MI 140 D=0:L=TL:NS=0:FOR C=1 TO
       NC:I=SN(C):L=L*LN(I):J=SN
       (C-1):NS=NS+SD(J):IF NS M
       OD 2 = 1 THEN D=D+12-RD(I
       ):GOTO 160
GE 150 D=D+RD(I)
EG 160 D = D MDD 12
OL 170 NEXT
ON 180 X=X+1.33*L*DX(D):Y=Y-.5*L
       *DY(D):LINE -(X,Y),1:SN(N
       C)=SN(NC)+1:FOR C = NC TO
        1 STEP -1:IF SN(C) <> 7
       THEN 200
OG 190 SN(C)=0:SN(C-1)=SN(C-1)+1
       :NEXT
AH 200 IF SN(0)=0 THEN 140
KE 210 IF INKEY$="" THEN 210
PO 220 GDTO 60
AF 230 DATA 0,0,1,0,1,7,0,10,0,0
       ,0,2,1,2
```

Commodore ML Saver

Buck Childress

This short, useful program saves any machine language program directly from memory into a disk or tape file. It works on any Commodore 64 or 128 (in 64 mode).

There are many useful machine language (ML) utilities available in public domain collections, on computer bulletin boards, and in publications like COMPUTE!. The most common way to place an ML program in memory is with a BASIC loader—a BASIC routine which READs the necessary values from DATA statements and POKEs them into memory. That method is fine for short ML programs, but can involve quite a delay when the ML program is long. It takes time, first of all, to load the BASIC loader. Then there's another wait while it POKEs everything into memory.

A much faster technique is to load the ML from disk or tape directly into memory. The only problem is making the tape or disk file. Machine language monitors such as Supermon can save any ML program directly from memory. But if you don't have a monitor, or don't know how to use one, that's not a viable option, either.

A Better Way

"ML Saver" is a short BASIC utility that can save any machine language program on disk or tape directly from where it resides in memory. After you type in and save ML Saver, load and run the program that creates the ML code you want to save. Make a note of the starting and ending addresses. If the ML is POKEd into memory with a FOR-NEXT loop, these addresses usually appear in the loop itself. For instance, if the loop is FOR J=49152 to 51000:READ Q:POKE J,Q:NEXT then you know the starting address is 49152 and the ending address is 51000. Now check the SYS address used to activate the program. This is usually the same as the starting address (for example, SYS 49152), but some programs are activated by jumping to an address somewhere in the middle of the code.

Once you have this information, you're ready to load and run ML Saver. The program asks you for the name you want to save the ML program under: Enter any name up to eight characters in length (extra characters are ignored). After you've supplied the name, enter D to save to disk or T for tape. Then enter the starting and ending addresses you wrote down earlier. ML Saver proceeds to save the ML code.

After the file has been created, you can load it with LOAD"*filename*",8,1 for disk or LOAD"*filename*",1,1 for tape (of course, you should replace *filename* with the filename you used when saving the program). Then SYS to the correct address to activate the program. To do this under program control, put the following statements at the beginning of your program:

```
10 IF J=1 THEN 30
20 J=1:LOAD"filename",8,1
30 REM PROGRAM CONTINUES HERE
```

When you run a program containing these lines, the variable J equals 0, so the computer falls through the IF test in line 10 and performs line 20. This line sets J to a nonzero value and loads the ML. After the load is complete, the computer automatically reruns the program beginning at line 10, but does not erase previously established variables. This time around, J equals 1, so the computer skips line 20 and proceeds with line 10. ©

Commodore ML Saver

For instructions on entering this listing, please refer to "The New Automatic Proofreader for Commodore" in this issue of COMPUTE!

```
FE 10  INPUT"{CLR}{DOWN}PROGRAM
       NAME";PN$:IFLEN(PN$)>8T
       HENPN$=LEFT$(PN$,8)
JC 40  FORJ=1TOLEN(PN$):POKE203
       9+J,ASC(MID$(PN$,J,1)):N
       EXTJ
AM 50  PRINT"{DOWN}{RVS}D{OFF}I
       SK OR {RVS}T{OFF}APE? ";
GH 60  GETA$:IFA$=""THEN60
HC 70  IFA$="D"THENDEVICE=8:GOT
       O100
EC 80  IFA$="T"THENDEVICE=1:GOT
       O100
FH 90  GOTO60
EC 100 PRINTA$:POKE780,15:POKE
       781,DEVICE:POKE782,255:
       SYS65466
DB 110 POKE780,LEN(PN$):POKE78
       1,248:POKE782,7:SYS6546
       9
FK 120 INPUT"{DOWN}BEGINNING A
       DDRESS";BA
XA 130 HI=INT(BA/256):LO=BA-(H
       I*256)
MS 140 POKE251,LO:POKE252,HI
PP 150 INPUT"{DOWN}ENDING ADDR
       ESS";EA
SJ 160 HI=INT(EA/256):LO=EA-(H
       I*256)+1
PF 170 POKE780,251:POKE781,LO:
       POKE782,HI
RF 180 PRINT"{DOWN}SAVING ML V
       ERSION OF ";PN$
KX 190 SYS65496:END                ©
```

Loading And Linking Commodore Programs

Part 1

Jim Butterfield, Associate Editor

This series covers the ins and outs of loading, chaining, and overlaying programs on Commodore computers.

The LOAD command seems easy enough to understand: It brings a program into memory from disk or tape. But it has special features and pitfalls. And when you cause one program to load another program, you enter the special field of chaining, overlaying, and bootstrapping. Let's take a close look at all of these operations, beginning with the LOAD command.

LOAD performs some subtle tasks, including relocating a program if necessary and a delicate job called relinking. There are special rules that apply when you load a program that was saved on a different type of Commodore computer. In fact, it sometimes works better to forget about LOAD and use a different technique. And the LOAD command behaves differently when executed by a program than it does when you enter it directly from the keyboard.

Most of the principles we'll cover in this series apply to programs stored on both disk and tape. It's easier to give examples that apply to disk systems, mostly because of the simplicity of setting up demonstration files. But you can still learn a lot about LOAD even if you don't have a disk drive.

What LOAD Does

When a LOAD command is executed, the following things happen:

- A PRG (PRoGram-format) file is brought into memory. *Program* is just the name for a certain type of file. The material contained in the file doesn't need to be a program. It could be a block of data, a screen, a character set, or anything.
- If the LOAD command doesn't specify nonrelocation, the information is normally *relocated*. That is, no matter what memory location it was saved from, it's loaded into memory beginning at the start of BASIC program space. PET/CBM computers are an exception to this rule: They never relocate programs. There's also a special cassette format available on VIC-20 and Commodore 64 computers which forces nonrelocation. (Unfortunately, this format is not easy for the beginner to create.)
- If the LOAD command specifies nonrelocation, the information is placed in memory at the same addresses from which it was saved. This is generally done by adding ,1 at the end of the LOAD command. For example, LOAD "PROG",8 loads the file PROG from disk and relocates it in memory, but LOAD "PROG",8,1 loads without relocating.
- If there were no errors during the load, the reserved variable ST is set to 0 (for tape) or 64 (for disk). It's often a good idea to check ST after loading. You can't rely on BASIC to catch every conceivable load error, particularly with tape. If you're using a disk drive, it's a good idea to check the drive status or at least see if the red error light is flashing.
- After the load is complete, the program is relinked (more about relinking in a moment).
- If the LOAD command was issued in direct mode (from the keyboard), all variables are cleared and the BASIC start-of-variables pointer is set to the first byte past the last byte that was loaded.
- If the LOAD command was issued by a program, variables are not cleared and the program automatically reruns from the beginning. This creates powerful opportunities, but requires some special handling to work correctly.

After a load is finished, what ends up in the computer's memory isn't quite the same as what you originally typed on the keyboard. For one thing, keywords are *tokenized*—compressed into single-byte values. When you type in the letters P-R-I-N-T, the BASIC word PRINT is "crunched" together into a single byte which the computer recognizes as PRINT. And the picture is further complicated because different Commodore computers use slightly different tokenizing schemes (we'll return to this point a bit later).

Relinking

When a BASIC program is in memory, each program line is linked to the next line by a chain of pointers (often called *line links*). At the start of each line there's a two-byte pointer that shows where the next line starts. The end of the program

is marked by a pointer that consists of two zeros.

What's the purpose of the chain? When the computer runs a program, there are many times when it needs to find a line number quickly—to execute a GOTO or GOSUB, for example. Rather than wade through every character of every line, it can skip from one link to the next until it finds the line it needs.

Each two-byte pointer shows the actual memory address where the next line begins, and these links are saved with the program. After a relocating load, which may bring the program into a different memory area, the pointers may point to the wrong locations. To fix everything up, the computer automatically *relinks* every line in the program after loading it into memory.

Relinking is a simple job. The computer scans through each program line, looking for the zero byte that marks the end of that line. As soon as it finds this address, the computer knows where the next line begins—at the next byte past the first line's end. It then rewrites the link for the first line and leaps ahead to repeat the process.

Relinking Problems

Two things can go wrong during the relinking process. If you load a chunk of data that doesn't contain any zero bytes, the computer can't find any end-of-line markers and won't be able to relink at all. The second difficulty is fortunately quite obscure: There's a model of computer called the B system in North America and the 700 series in Europe. The most well-known model in the USA is the B128 (not to be confused with the Commodore 128), but the comments here apply to all B and 700 models. When you save a BASIC program from these computers, the program is stored in an unusually low memory address. Because the saved program contains zeros in unexpected places, other Commodore computers can't make any sense of the chain.

Let's create an unlinkable file. If you have a disk drive, enter the following statements in direct mode (without line numbers), pressing RETURN after each line:

```
OPEN 8,8,8,"0:CRASHER,P,W"
PRINT#8,CHR$(1);CHR$(4);
FOR J=1 TO
300:PRINT#8,CHR$(4);:NEXT
CLOSE 8
```

Don't forget to put a semicolon at the end of each of the two PRINT# statements.

IF you LOAD "CRASHER",8 the computer prints LOADING and READY, but the cursor doesn't return. It's stuck in an endless loop, looking for the nonexistent zero that marks the end of the first BASIC line. This kind of crash is fairly common on cassette systems, when a bad load fills memory with garbage. It can also occur with disk if you forget to add ,1 to a LOAD command that requires relocation, loading something odd like a screen image into the BASIC program area. You can usually recover control by pressing RUN/STOP–RESTORE and entering NEW.

Before we look at a solution to the second problem, let's talk about another area where incompatibility arises between machines.

LOAD Address Incompatibility

PET/CBM computers can't relocate programs at all. On a PET/CBM, BASIC program space starts at location 1025. Most other Commodore computers use a different address, meaning that the PET/CBM can't load programs saved on those computers. The program following this article is a converter to solve both of these problems. You'll need a disk drive to use it.

The new file created by the program is loadable by any eight-bit Commodore computer. Since it sets the load address to 1025, PET/CBM computers can load it properly (all other models relocate it automatically). To make B128 program files usable by other computers, it puts dummy link bytes (both containing a 1) at the beginning of each line. Remember, all these computers relink programs automatically, so this problem can be solved by putting any nonzero values in the links.

This program works only with BASIC programs, since it stops at the two zero bytes that mark the end of the program. If there's something more "pasted" onto the end of the BASIC text—a machine language routine, for example—it will not be copied.

Alternative Method

If you have access to the type of computer which generated the original program, there's an easier way to do the same thing. You can make a Commodore 64 emulate the memory configuration of another machine so BASIC programs are kept in a more compatible part of memory. If you move the start of BASIC to location 1025, the links will be more conventional and the program will be PET-compatible.

To make the area at 1025 available for BASIC, we'll also need to relocate the screen to a new area. To fit the following commands onto two screen lines, you can abbreviate PRINT as ? and POKE as P SHIFT-O:

```
POKE56576,5:POKE 53272,4:POKE648,128
 :POKE1024,0:POKE44,4:POKE56,128
 :PRINTCHR$(147):NEW
```

The first three POKEs move the screen. The next three move the start and end of the BASIC area, and the last two commands clear the new screen and set up the new BASIC work area.

Once you've entered the above commands, your Commodore 64 emulates a PET/CBM's BASIC configuration. You may now convert a BASIC program into compatible format by loading it into the reconfigured machine and saving it again. By using this load/save sequence, you're making several things happen. When you load the program, it's relocated into a new area. The chain links are then rebuilt to be compatible with the new memory space. When you save, the newly relocated program—complete with new links—is placed on disk or tape.

The modified program seems the same, but it now has a "universal" style. Whether you created it with the Converter program below or with the POKEs and LOAD/SAVE sequence, it can now be loaded into any eight-bit Commodore machine. Its addresses are directly compatible with the PET/CBM, and other machines will relocate the program as it loads. And we've eliminated the possibility of the peculiar B128 chain that confuses other Commodore computers.

Incompatible Tokens

Some programs won't transfer from one machine to another because of differences in the BASIC tokens. You'll have little trouble with commonly used commands such as PRINT and IF. The difference comes with more advanced commands that aren't part of every version of Commodore BASIC. If you write a program in CBM BASIC 4.0 and use commands like DLOAD and SCRATCH, don't be surprised to find that it doesn't run properly on a Commodore 64. Those commands don't exist in the 64's BASIC.

You might also be surprised to find that a Plus/4 or Commodore 128 (in 128 mode) can't run the PET/CBM program either, even though both of those machines have DLOAD and SCRATCH commands. Why not? The commands are there, but they're represented by different token values.

In such cases, you can't use LOAD and SAVE at all. What you'll have to do is detokenize the program by LISTing it to a disk file, then bring it back into memory with a "merge" method like the one described in "Commodore Dynamic Keyboard, Part 3" (COMPUTE!, December 1985).

We've only scratched the surface of the many uses of the LOAD command. When we start to look into chaining, overlaying, and reloading techniques, we'll discover that the LOAD command has amazing potential.

Commodore Program Converter

For instructions on entering this listing, please refer to "The New Automatic Proofreader for Commodore" published in this issue of COMPUTE!

```
CA 110 OPEN15,8,15
QA 120 INPUT "NAME OF PROGRAM"
       ;N$
JA 130 OPEN 2,8,2,"0:"+N$+",P,
       R"
KJ 140 INPUT#15,E,E$,E1,E2:IF
       {SPACE}E THEN PRINT E$:
       STOP
JC 150 INPUT"NAME OF CONVERTED
       PROGRAM";C$
BG 160 OPEN3,8,3,"0:"+C$+",P,W
       "
RF 170 INPUT#15,E,E$,E1,E2:IF
       {SPACE}E THEN PRINT E$:
       STOP
JQ 180 Z$=CHR$(0)
FA 190 REM:READ LOAD ADDRESS
FX 200 GET#2,A$,B$
DC 210 PRINT#3,CHR$(1);CHR$(4)
       ;
SR 220 REM:READ CHAIN
CB 230 GET#2,A$,B$
SS 240 IF LEN(A$)+LEN(B$)=0 GO
       TO 370
GA 250 PRINT#3,CHR$(1);CHR$(1)
       ;
CH 260 REM: READ LINE NUMBER
QG 270 FOR J=1 TO 2
HK 280 GET#2,A$:IF A$="" THEN
       {SPACE}A$=Z$
HQ 290 PRINT#3,A$;
HJ 300 NEXT J
FE 310 REM:READ LINE
JP 320 GET#2,A$:IF A$="" THEN
       {SPACE}A$=Z$
ES 330 PRINT#3,A$;
SD 340 IF A=Z$ GOTO 230
DJ 350 GOTO 320
QB 360 REM:WIND UP FILES
EJ 370 INPUT#15,E,E$,E1,E2: IF
       E THEN PRINT E$:STOP
AX 380 PRINT#3,Z$;Z$;
JF 390 CLOSE 3
JE 400 CLOSE 2
FF 410 CLOSE 15
```

©

Atari P/M Graphics Toolkit

Tom R. Halfhill, Editor

If you're mystified by Atari player/missile graphics, this article is for you. With help from a toolkit of routines written in BASIC and machine language, you'll soon be writing your own programs that move shapes of your own design anywhere on the screen quickly and easily. For all 400/800, XL, and XE computers with Atari BASIC.

It's safe to say that no feature on Atari computers is as exciting and as frustrating as player/missile graphics.

Exciting: P/M graphics provides four larger objects called players and four smaller objects called missiles which can be displayed in almost any shape or color and animated on the screen. You can even achieve 3-D effects by making the objects pass above and beneath background graphics and each other.

Frustrating: Atari BASIC has no special commands for using P/M graphics, standard Atari manuals don't cover P/M graphics, you can't set up P/M graphics in a program without worrying about lots of POKEs and protecting memory, and Atari BASIC lacks a straightforward way to move a P/M object vertically or diagonally at speeds faster than a crawl. Furthermore, there's no easy method of designing player shapes without scribbling on graph paper and adding up binary bit values of bytes.

"Atari P/M Graphics Toolkit" solves all these problems and more. It's a package of routines written in BASIC and machine language that can form the core of your own programs. With the Toolkit, you can easily design your own shapes with a joystick, then write simple BASIC programs that automatically set up P/M graphics and instantly move your objects anywhere on the screen.

You've probably used or heard of similar programs in other magazines and books. In fact, several popular routines for animating P/M graphics appeared in early issues of COMPUTE! and are reprinted in various COMPUTE! books. But the P/M Graphics Toolkit offers these advantages:

Special Features

- The Toolkit setup/animation routine creates a true X-Y coordinate system for moving P/M objects to any horizontal-vertical position on the screen. This system is patterned after the X-Y coordinates in Atari BASIC graphics modes, so if you know how to use the PLOT, DRAWTO, LOCATE, or POSITION commands, you should have no trouble animating P/M objects with the Toolkit.
- A machine language subroutine automatically clears out the P/M memory area in a flash, so your programs initialize faster.
- The Toolkit routines work in single-resolution or double-resolution P/M graphics modes, and in any screen graphics mode.
- Unlike most other programs of this type, the Toolkit lets you move any of the four missile objects as easily as any of the four players.
- The Toolkit allows you to instantly change the shape or size of a P/M object by selecting from any number of previously designed shapes—even while the object is moving.
- Because all of its machine language is written to be completely relocatable, you can add the Toolkit setup/animation routine to any BASIC program without fear of memory conflicts with other ML routines you may be using.
- Best of all, using the Toolkit is a snap. Once the Toolkit setup/animation routine is added to your program, initializing P/M graphics requires only one line of BASIC, and all of the animation and shape-flipping can be done with just a single BASIC statement.

Getting Started

All of these features are contained in the setup/animation routine listed below as Program 1. This will be the basic building block of your own programs, so the line numbers start at 20000 to leave plenty of room for your own lines.

Be sure to type in Program 1 using COMPUTE!'s "Automatic Proofreader" utility, because the DATA statements in lines 20170–20590 are extremely critical—they encode the machine language for two ML subroutines. When you're done, store Program 1 on disk or tape with the LIST command, not SAVE or CSAVE:

LIST"D:*filename.ext***"** for disk
LIST"C:" for cassette

This way, you can use the ENTER command (ENTER"D:*filename.ext*" or ENTER"C:") to merge the routine with another program already in memory, as we'll demonstrate in a moment.

A Single-Line Setup

To create a program that uses player/missile graphics, you only have to call this routine once. The call should be as near to the beginning of your program as possible; the first line is ideal. Here's the proper format:

```
10 GRMODE=0:PMMODE=1:GOSUB
   20050
```

Set the variable GRMODE to whatever graphics mode you desire. Any valid number or expression you'd use with the GRAPHICS statement will work. In the above example, we're asking the Toolkit routine to set up GRAPHICS 0. Set the variable PMMODE to the player/missile graphics mode you want: either 1 for single-line resolution or 2 for double-line resolution. (If you aren't familiar with P/M modes, see "Atari Animation with P/M Graphics," a three-part series beginning in the September 1985 issue of COMPUTE!. Even though the Toolkit routines automatically perform the memory allocation and animation chores discussed in this series, I strongly recommend acquiring some background on P/M graphics.)

The third statement in the sample line above actually calls the Toolkit setup/animation routine. Notice that it GOSUBs to line 20050 instead of 20000. Lines 20000–20040 are REM statements, and it's good programming practice to avoid a GOSUB or GOTO reference to a REM because some people delete REM statements from programs to save memory.

When you call the routine in this manner, there will be a pause of several seconds as it loads the machine language into memory. Then, in rapid sequence, the routine automatically protects the right amount of memory for the P/M mode you requested; instantly clears out any old data in that memory area; performs all the POKEs necessary to set up P/M graphics; assigns colors to all four players and missiles; and switches to the graphics mode you requested. In other words, it does all of the dirty work for you.

When the Toolkit routine is finished, a RETURN statement passes control back to whatever line follows the GOSUB. That's where the rest of your program should continue. Be sure to place an END statement at the end of your own program lines, but before line 20000, so the Toolkit routine doesn't accidentally execute more than once.

If you want to change the player colors from the colors assigned by the Toolkit routine, POKE your own color values into locations 704–707.

PMMOVE Magic

After this simple setup, all it takes is one BASIC statement to move any player or missile anywhere on the screen. Here's the format:

```
A=USR(PMMOVE,PLAYER#,SHAPE
,SIZE,X,Y)
```

Let's take a look at these parameters one at a time. We'll follow with a few examples.

The variable A is a dummy variable required by the USR statement—with this routine, it returns no useful value. PMMOVE is the address of the machine language subroutine, and its value is automatically set when your program executes the GOSUB 20050 described above. (PMMOVE is actually the address of PMMOVE$, a string which holds the ML data.) Don't change the value of PMMOVE unless you enjoy watching your computer crash.

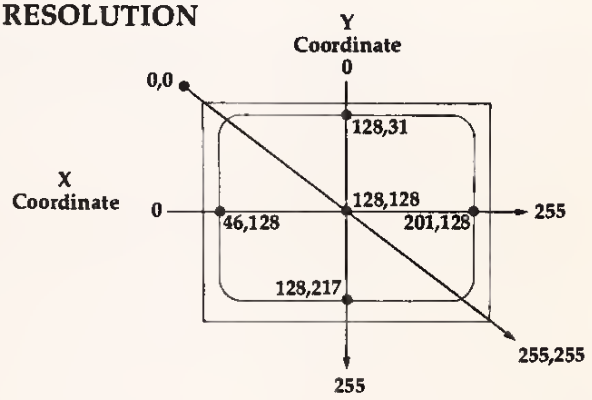
The remaining five parameters are under your control. You must assign values to these parameters yourself and always include them when calling the PMMOVE routine.

PLAYER# should be a number from 1 to 8 that specifies which P/M object will be affected by the statement. Numbers 1 to 4 specify the four players, and numbers 5 to 8 specify the four missiles.

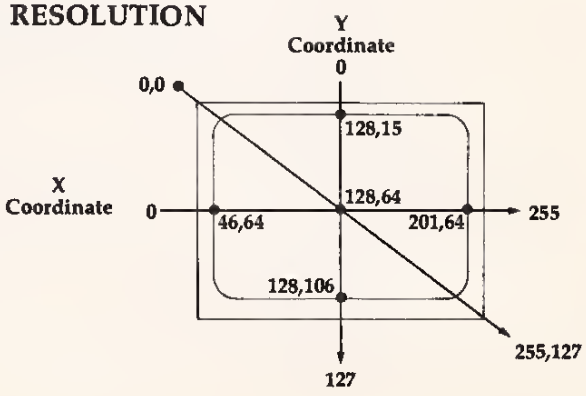
SHAPE is the address of the shape data for the player or missile. The best way to use this parameter is to specify the address of a string which contains previously created shape data. For background on designing player shapes, see Part 2 of the three-part series mentioned above. In a moment, we'll discuss a Toolkit utility that lets you design your own shapes with a joystick and which calculates the shape data for you. The SHAPE parameter, incidentally, is what allows players to flip between different shapes even

PMMOVE SCREEN COORDINATES

SINGLE RESOLUTION



DOUBLE RESOLUTION



while they're moving. Each time you move a player or missile, the PMMOVE subroutine erases the object's entire memory area and replaces it with the new image you select with the SHAPE parameter. If you don't quite follow this explanation, the example programs will clarify things.

SIZE is the height of the player or missile in bytes. If the shape data referred to by SHAPE consists of eight bytes, you'd insert an 8 for this parameter. This makes it possible to store the data for numerous player shapes in a single string, then select just the shape you want by pointing SHAPE to its position within the string and specifying the substring's length with SIZE. Again, the example programs will demonstrate this technique.

The final two parameters determine the new position of the player or missile on the screen. X is the horizontal coordinate and Y is the vertical coordinate. Like the screen coordinates used by BASIC graphics commands such as PLOT and DRAWTO, position 0,0 refers to the upper-left corner. As the X coordinate increases, the P/M object moves from left to right; as the Y coordinate increases, the object moves from top to bottom. X values can range from 0 to 255. Y values can range from 0 to 255 in single-resolution P/M graphics or 0 to 128 in double-resolution P/M graphics. If the X or Y values exceed these ranges, the object eventually "wraps around" to the opposite side of the screen. But if you specify an X or Y value which is less than 0, an error results.

The accompanying figure shows the layout of these coordinates. Notice how some positions are off the visible screen area. A quick way to make an object disappear is to move it to one of these "invisible" positions. (Another way is to specify 0 for the SIZE parameter.)

Frame Flipping

Now for the fun stuff. The following examples show some typical ways to use the PMMOVE statement in your own programs.

Program 2 demonstrates how to store a player shape in a string. After typing Program 2, LIST it to disk or tape, then merge it in memory with Program 1 by using the ENTER command. When you type RUN, there's a pause as everything sets up, then a player in the form of a smiling face appears in the center of the screen. Here's how it works:

Line 10 calls the Toolkit setup routine (Program 1), specifying single-resolution P/M and graphics mode 0. Line 20 dimensions the string variable SHAPE$ to hold 11 characters, the size of the player (11 bytes tall). Next it uses BASIC's ADR function to set the variable SHAPE to the address of SHAPE$, and sets SIZE to 11. Then it uses a FOR-NEXT loop to read the 11 bytes of shape data in line 40 into SHAPE$.

Finally, the PMMOVE statement in line 30 makes the player appear at position 127,127 (the centerpoint in single-res P/M graphics). Note that the PLAYER# parameter is 1; simply by changing this number to 2, 3, or 4, you can display any of the other players using the same shape data—try it. (By the way, you should press SYSTEM RESET before rerunning this or any other program that uses player/missile graphics. Otherwise, the reserved P/M memory keeps growing until it consumes all the RAM in your computer.)

The technique of storing player shape data in strings has some interesting applications. By storing several shapes in a single string and flipping rapidly between them, you can make your players come alive as they move about the screen. Even a static player can seem to move as its shape continuously changes, much like frame animation in a cartoon. To see an example, type in Program 3. LIST it to disk or tape, then merge it in memory with Program 1 using the ENTER command. When you type RUN, the program initializes for a few seconds, then displays a small explosion in the middle of the screen, hurling fragments in all directions. Yet, throughout this animated sequence, the player object never moves.

The secret is BOOM$, a 64-character string filled with eight player shapes in line 30. Each shape is eight characters long, as seen in the DATA statements in lines 90–160. The PMMOVE statement in line 60 rapidly flips through all eight shapes in sequence because it is sandwiched within the FOR-NEXT loop between lines 40 and 80. (The loop serves double duty; it also fades the explosion sound into silence.) Another FOR-NEXT in line 70 is a simple delay loop. To slow down the explosion sequence for a better idea of what's going on, change the 35 in line 70 to a larger number—say, 150. Also notice how the final player shape in this sequence consists of nothing but zeroes. This is another way of making the player object seem to disappear without moving it off the screen.

Whirling Dervishes

Another fascinating application of this technique is to design players which change shape depending on which direction they're moving. To see an example, type in Program 4. As before, LIST the program to disk or tape, then merge it with Program 1 using ENTER.

When you type RUN, a red tank appears in the center of the screen. The low rumble of an idling engine can be heard in the background. Plug a joystick into port 1, then try moving the stick right or left. Notice how the tank rotates clockwise or counterclockwise. In fact, it spins so fast it's almost a blur.

The player isn't actually rotating or moving, of course; it's simply changing shape many times per second, thanks to the SHAPE parameter of the PMMOVE routine. The secret, again, is a string (SHAPE$) which contains shape data for eight tank images, one for each of the eight possible directions. Lines 40–80 read the joystick and determine which image should be displayed.

Now push the joystick forward. The engine revs up and the tank obediently moves in the direction it is pointed. You can drive the tank all over the screen. To see how this works, study lines 90–190. Notice how the ON-GOTO statement in line 90 passes control to one of the lines from 120 to 190, depending on which direction the tank is oriented. Each of these lines either increments or decrements the X and

Y coordinates to move the tank in one of the eight possible directions.

Notice, too, how this entire program contains only the one PMMOVE statement in line 80—a single statement controls both the player's shape and its movement.

Stepping On The Gas

To make the tank move twice as fast, change lines 120–190 so the X and Y coordinates are incremented or decremented by 2 instead of 1. For still more speed, you can even change these values to 3. However, keep in mind that the movement is not quite as smooth as these step values are increased; the object seems to jerk along at higher speeds. A step value of 2 or 3 is a reasonable compromise between speed and loss of grace.

Try changing the PLAYER# parameter in the PMMOVE statement to 2, 3, or 4 to see the other players in action.

For an example of missile animation, merge Program 5 with both Program 1 and Program 4. These lines let the tank fire a shot when you press the joystick button. The direction routine is very similar to the one in Program 4; in fact, it uses the same variable (DIR) to figure out which way the tank is pointing.

Note how the missile shape defined in line 25 is only one byte long—CHR$(3). Unlike player objects, which are eight bits wide, missiles are only two bits wide. You can't design a very fancy shape with only two bits to work with, so most programs use the missiles for such tiny shapes as projectiles, bullets, etc. A single byte is enough to define a missile shape for these purposes. Since all four missiles share the same memory area as a single player, defining missile shapes is a little different than defining player shapes.

Rather than getting into a confusing discussion about missiles and bit manipulation, just remember this: The shape bytes for the four missiles (accessed as PLAYER# 5, 6, 7, and 8 in the PMMOVE statement) should be 3, 12, 48, and 192, respectively. For instance, the PMMOVE statements in lines 1100 and 1110 refer to the first missile, PLAYER# 5, so its shape data in MISSILE$ is a 3. If you want to move the second missile, change the PLAYER# parameter to 6 and the shape byte in line 25 to CHR$(12). If you want to move the third missile, change the PLAYER# parameter to 7 and the shape byte to CHR$(48). And if you want to move the fourth missile, change the PLAYER# parameter to 8 and the shape byte to CHR$(192). These shape bytes will work with missiles in any of your own programs.

Designing Player Shapes

As a final touch, the P/M Graphics Toolkit includes a small utility for designing your own players. It's not very elaborate, but it's better than scribbling on graph paper and counting up "on" bits and "off" bits in your head.

Like the other example programs, the shape utility is based on the Toolkit setup/animation routine. To prepare it, type in Program 6 and merge it with Program 1. SAVE or CSAVE a copy of the merged program so you won't have to repeat these steps each time you want to use the utility.

After typing RUN, select single- or double-resolution player/missile graphics by pressing 1 or 2. The utility spends a few moments initializing, then displays a grid of dots along the left side of the screen. A cursor is at the upper-left corner; it's controlled by a joystick plugged into port 1.

The grid represents a magnified view of an eight-bit-wide player strip, with one dot for each bit. Each horizontal row of eight dots, therefore, represents one byte of the player strip. Player strips are really 255 bytes tall in single resolution or 128 bytes tall in double resolution, but there isn't room to display a grid that large on the screen. Still, the grid is tall enough to design player shapes for most purposes.

To design a shape, just move the cursor anywhere on the grid and press the joystick button to set a bit. The dot changes to inverse video, and an actual-size player begins taking form in the blank area on the right side of the screen. Simultaneously, a number representing the byte value for that row of bits appears. Try moving the cursor and setting more bits; the byte values keep changing. To erase a bit that you've set, position the cursor on it and press the joystick button again. With each change, the byte values are updated along with the actual-size player.

When you're satisfied with a player shape, jot down the byte values. (You can ignore the zeroes, if any, above and below the shape.) These byte values represent the shape data for your player. To display the player in your own programs, just read the numbers into a string as demonstrated in Programs 2, 3, and 4. Use the address of this string as the SHAPE parameter in the PMMOVE statement. The number of shape bytes becomes the value for the SIZE parameter.

If you mess things up too badly when designing a player, you can erase the entire grid by pressing the E key.

This is a bare-bones utility, but it's enough to get you started and take the bothersome paperwork out of defining player shapes. If you want, you can add more features of your own—commands to change player colors; to shift bit patterns left, right, up, or down; to flip the pattern as a mirror image; to automatically generate DATA statements of player shape bytes; and so on.

A Bonus Routine

Both machine language subroutines used by the Toolkit are designed to be as crashproof as possible. If you accidentally pass the wrong number of parameters in a USR statement, the routines immediately clear all faulty values off the 6502 stack and bounce back to BASIC. So if you call the PMMOVE routine and nothing happens, check the USR statement to make sure you included all of the parameters and that the parameters have legal values.

You may find one of the Toolkit machine language routines useful in other types of programs as well. PAGCLR is a general-purpose routine that rapidly clears a specified number of memory pages with zeroes. For P/M graphics, this routine erases any garbage data that may be cluttering the reserved memory area. But it's handy for any program that needs to clear out a

large amount of memory in a split-second.

The ML data for PAGCLR can be found in the DATA statements in Program 1 at lines 20160–20240. Line 20060 reads this data into the string PAGCLR$, previously DIMensioned to 48 characters. The variable PAGCLR is set to the address of PAGCLR$.

Here's the format for calling the PAGCLR routine:

```
A=USR(PAGCLR,ADDRESS,PAGES)
```

where PAGCLR is the address of the ML routine, ADDRESS is the starting address (in decimal) of the memory area you want to clear, and PAGES is the number of memory pages to clear (a page equals 256 bytes). For example, the last statement in line 20130 clears either 1,024 or 2,048 bytes starting at the memory address PMBASE, depending on whether the variable PAGES is set to 4 or 8 for double- or single-resolution P/M graphics.

Caution: Don't use the PAGCLR routine for other purposes unless you understand exactly how it works. If misdirected, it can wipe out massive amounts of memory in an instant and crash your computer.

For instructions on entering these listings, please refer to "COMPUTE!'s Guide to Typing In Programs" published in this issue of COMPUTE!.

Program 1: P/M Toolkit Setup Routine

```
HF 20000 REM *** PLAYER/MISS
         ILE SETUP ***
DD 20010 REM DEFINE PMMDDE &
          GRMDDE BEFDRE CALL
         ING THIS RDUTINE
JE 20020 REM PMMDDE=1 FDR SI
         NGLE-RES P/M
IP 20030 REM PMMDDE=2 FDR DD
         UBLE-RES P/M
EL 20040 REM GRMDDE=GRAPHICS
          MDDE
GE 20050 DIM PAGCLR$(48),PMM
         DVE$(202):PAGCLR=AD
         R(PAGCLR$):PMMDVE=A
         DR(PMMDVE$)
HL 20060 MT=0:RESTDRE 20170:
         FDR X=1 TD 48:READ
          A:PAGCLR$(X)=CHR$(A
         )
DA 20065 MT=MT+A:NEXT X:IF M
         T=7484 THEN MT=0:GD
         TD 20070
KO 20066 PRINT "ERRDR IN DAT
         A... LINES 20170-20
         240":STDP
GF 20070 FDR X=1 TD 202:READ
          A:PMMDVE$(X)=CHR$(
         A):MT=MT+A:NEXT X
JE 20075 IF MT<>23367 THEN P
         RINT "ERRDR IN DATA
         ... LINES 20260-205
         90":STDP
DF 20080 IF PMMDDE=1 THEN PA
         GES=8:DMA=62:GDTD 2
         0110
DF 20090 IF PMMDDE=2 THEN PA
         GES=4:DMA=46:GDTD 2
         0110
ND 20100 RETURN
MK 20110 PDKE 54279,PEEK(106
         )-PAGES:PDKE 106,PE
         EK(106)-PAGES:PDKE
         207,PMMDDE
AE 20120 GRAPHICS GRMDDE
DM 20130 PMBASE=PEEK(106)*25
         6:PDKE 559,DMA:PDKE
          53277,3:X=USR(PAGC
         LR,PMBASE,PAGES)
PN 20140 PDKE 704,68:PDKE 70
         5,78:PDKE 706,88:PD
         KE 707,98:REM P/M C
         DLDRS
NI 20150 RETURN
EJ 20160 REM PAGCLR ML DATA
ND 20170 DATA 104,201,2,240,
         16,133
CA 20180 DATA 206,162,0,228,
         206,208
PD 20190 DATA 1,96,104,104,2
         32,169
AM 20200 DATA 0,240,244,104,
         133,204
GN 20210 DATA 104,133,203,10
         4,104,133
CE 20220 DATA 205,169,0,168,
         170,145
HB 20230 DATA 203,200,208,25
         1,230,204
FL 20240 DATA 232,228,205,20
         8,244,96
BE 20250 REM PMMDVE ML DATA
DC 20260 DATA 104,201,5,240,
         18,141
DJ 20270 DATA 0,4,162,0,236,
         0
IK 20280 DATA 4,208,1,96,104
         ,104
BD 20290 DATA 232,169,0,240,
         243,104
LD 20300 DATA 104,201,9,144,
         9,104
GK 20310 DATA 104,104,104,10
         4,104,104
LA 20320 DATA 104,96,24,201,
         0,240
LA 20330 DATA 242,141,4,4,10
         4,133
HE 20340 DATA 206,104,133,20
         5,104,104
KP 20350 DATA 141,5,4,104,10
         4,141
EK 20360 DATA 2,4,104,104,14
         1,3
PH 20370 DATA 4,174,4,4,173,
         2
MG 20380 DATA 4,157,255,207,
         224,5
GC 20390 DATA 176,2,144,5,16
         9,0
LC 20400 DATA 141,4,4,165,20
         7,201
LI 20410 DATA 2,240,28,165,1
         06,24
FA 20420 DATA 105,3,109,4,4,
         133
CA 20430 DATA 204,169,0,133,
         203,168
HD 20440 DATA 145,203,200,20
         8,251,173
ID 20450 DATA 3,4,133,203,24
         ,144
LN 20460 DATA 65,165,106,24,
         105,1
IH 20470 DATA 133,204,169,12
         8,133,203
IJ 20480 DATA 173,4,4,240,21
         ,162
DN 20490 DATA 0,165,203,24,1
         05,128
GB 20500 DATA 133,203,165,20
         4,105,0
LF 20510 DATA 133,204,232,23
         6,4,4
GD 20520 DATA 208,237,160,0,
         152,145
IG 20530 DATA 203,200,192,12
         7,208,249
LK 20540 DATA 173,3,4,201,12
         8,144
OF 20550 DATA 1,96,101,203,1
         33,203
GI 20560 DATA 165,204,105,0,
         133,204
FA 20570 DATA 160,0,204,5,4,
         240
CE 20580 DATA 8,177,205,145,
         203,200
KF 20590 DATA 24,144,243,96
```

Program 2: Player Shape Demo

```
KH 10 PMMDDE=1:GRMDDE=0:GDSU
      B 20050
JL 20 DIM SHAPE$(11):SHAPE=A
      DR(SHAPE$):SIZE=11:RES
      TDRE 40:FDR X=1 TD 11:
      READ A:SHAPE$(X)=CHR$(
      A):NEXT X
DN 30 A=USR(PMMDVE,1,SHAPE,S
      IZE,127,127)
MG 40 DATA 24,60,126,90,219,
      255,219,195,102,60,24
OM 50 END
```

Program 3: Explosion Animation Demo

```
KI 10 PMMDDE=2:GRMDDE=0:GDSU
      B 20050
FA 20 SETCDLDR 2,0,0:PDKE 70
      4,72
IN 30 DIM BDDM$(64):BDDM=ADR
      (BDDM$):RESTDRE 90:FDR
       X=1 TD 64:READ A:BDDM
      $(X)=CHR$(A):NEXT X
DD 40 FDR VDLUME=14 TD 0 STE
      P -2
CI 50 SDUND 0,90,0,VDLUME:SD
      UND 1,100,4,VDLUME
KN 60 A=USR(PMMDVE,1,BDDM,8,
      127,64):BDDM=BDDM+8
BO 70 FDR DELAY=1 TD 35:NEXT
       DELAY
HP 80 NEXT VDLUME
LL 90 DATA 0,0,8,28,28,8,0,0
EK 100 DATA 0,8,34,92,20,34,
       8,0
IL 110 DATA 8,65,4,168,20,1,
       64,8
AB 120 DATA 148,1,20,160,1,2
       0,1,136
IE 130 DATA 145,74,32,130,65
       ,2,84,137
HO 140 DATA 72,1,64,0,130,1,
       8,82
HI 150 DATA 129,0,0,0,0,128,
       1,66
GF 160 DATA 0,0,0,0,0,0,0,0
GP 170 END
```

Program 4: Tank Animation Demo

```
KI 10 PMMODE=2:BRMODE=0:BOSU
      B 20050
AL 20 DIM BHAPE$(64):BIZE=8:
      RESTORE 210:FOR X=1 TO
      64:READ A:SHAPE$(X)=C
      HR$(A):NEXT X
HH 30 DIR=1:X=127:Y=64:BOUND
      0,180,6,3
DB 40 B=BTICK(0):IF B=11 THE
      N DIR=DIR-1
EB 50 IF B=7 THEN DIR=DIR+1
HE 60 IF DIR<1 THEN DIR=8
HH 70 IF DIR>8 THEN DIR=1
LB B0 A=USR(PMMOVE,1,ADR(SHA
      PE$(DIR*8-7)),BIZE,X,Y
      )
DA 90 IF S=14 THEN BOUND 1,1
      20,6,6:ON DIR GOTO 120
      ,130,140,150,160,170,1
      80,190
FH 100 IF B=15 THEN BOUND 1,
      180,6,3:GOTO 40
CP 110 GOTO 40
LH 120 Y=Y-1:GOTO 40
DL 130 X=X+1:Y=Y-1:GOTO 40
LF 140 X=X+1:GOTO 40
DL 150 X=X+1:Y=Y+1:GOTO 40
LJ 160 Y=Y+1:GOTO 40
DP 170 X=X-1:Y=Y+1:GOTO 40
LL 180 X=X-1:GOTO 40
ED 190 X=X-1:Y=Y-1:GOTO 40
CC 200 REM *** TANK SHAPES (
      CLOCKWISE N-NE-E-SE-S
      -SW-W-NW ***
LO 210 DATA 8,8,42,62,62,62,
      62,34
FD 220 DATA 9,26,60,127,254,
      60,24,16
BF 230 DATA 0,252,120,127,12
      0,252,0,0
FD 240 DATA 16,24,60,254,127
      ,60,26,9
MC 250 DATA 34,62,62,62,62,4
      2,8,8
IO 260 DATA 8,24,60,127,254,
      60,88,144
KK 270 DATA 0,63,30,254,30,6
      3,0,0
JA 280 DATA 144,88,60,254,12
      7,60,24,8
```

Program 5: Missile Demo

```
LF 25 DIM MISSILE$(1):MISSIL
      E$(1)=CHR$(3)
OD 85 IF BTRIG(0)=0 THEN GOS
      UB 1000
NB 1000 ON DIR GOTO 1010,102
        0,1030,1040,1050,106
        0,1070,1080
HL 1010 OX=0:OY=-2:GOTO 1090
HO 1020 OX=2:OY=-2:GOTO 1090
FA 1030 OX=2:OY=0:GOTO 1090
FD 1040 OX=2:OY=2:GOTO 1090
FC 1050 OX=0:OY=2:GOTO 1090
IC 1060 OX=-2:OY=2:GOTO 1090
IB 1070 OX=-2:OY=0:GOTO 1090
HE 1080 OX=-2:OY=-2
JP 1090 MX=X:MY=Y:FOR SHOOT=
        1 TO 15
BH 1100 A=USR(PMMOVE,5,ADR(M
        ISSILE$),1,MX,MY):MX
        =MX+OX:MY=MY+OY:NEXT
         SHOOT
HH 1110 A=USR(PMMOVE,5,ADR(M
        ISSILE$),0,0,0):RETU
        RN
```

Program 6: P/M Shapemaker

```
BF 10 GRAPHICS 2:SETCOLOR 2,
      0,0:POKE 752,1:POSITIO
      N 3,2:? #6;"P/M SHAPEM
      AKER"
KK 20 ? "1 SINGLE RESOLUTION
      ":? "2 DOUBLE RESOLUTI
      ON"
CD 30 OPEN #1,4,0,"K:":? "
      PRESS 1 OR 2";
PC 40 GET #1,A:IF A=49 THEN
      PMMODE=1:GOTO 70
DB 50 IF A=50 THEN PMMODE=2:
      GOTO 70
AD 60 GOTO 40
HJ 70 ? "{3 SPACES}PLEASE WA
      IT...":BRMODE=0:BOSUB
      20050
JL 80 DIM CURSOR$(2),OLD$(2)
      ,BIT(8),BYTE$(24):CURS
      OR$(1,2)="■{LEFT}":OLD
      $(1,2)=". {LEFT}"
BK 90 RESTORE 100:FOR N=1 TO
       8:READ A:BIT(N)=A:NEX
      T N
HP 100 DATA 128,64,32,16,8,4
       ,2,1
HH 110 POKE 752,1:BETCOLOR 2
       ,0,0:FOR N=1 TO 24:?
       ".....  0":BYTE$(N)
       =CHR$(0):NEXT N:CX=2:
       CY=0
EI 120 A=USR(PMMOVE,2,ADR(BY
       TE$),24,170,128/PMMOO
       E)
HH 130 POSITION 24,20:? "{RA
       8E PLAYER"
HE 140 LOCATE CX,CY,A:OLD$(1
       )=CHR$(A):POSITION CX
       ,CY:? CURSOR$;
FE 150 B=STICK(0):FOR N=1 TO
        25:NEXT N:IF S=14 TH
       EN 220
LC 160 IF S=7 THEN 240
OC 170 IF S=13 THEN 260
OD 180 IF B=11 THEN 280
BC 190 IF STRIG(0)=0 THEN 300
AK 200 IF PEEK(764)=42 THEN
       POKE 764,255:? CHR$(1
       25):GOTO 110
BC 210 GOTO 150
PB 220 IF CY=0 THEN 150
CH 230 ? OLD$;:CY=CY-1:GOTO
       140
PL 240 IF CX=9 THEN 150
CF 250 ? OLD$;:CX=CX+1:GOTO
       140
CJ 260 IF CY=22 THEN 150
CJ 270 ? OLD$;:CY=CY+1:GOTO
       140
PI 280 IF CX=2 THEN 150
CL 290 ? OLD$;:CX=CX-1:GOTO
       140
OA 300 IF OLD$(1)="■" THEN O
       LD$(1)=".":BYTE$(CY+1
       ,CY+1)=CHR$(ASC(BYTE$
       (CY+1))-BIT(CX-1)):GO
       TO 320
OF 310 OLD$(1)="■":BYTE$(CY+
       1,CY+1)=CHR$(ASC(BYTE
       $(CY+1))+BIT(CX-1))
BQ 320 POSITION 11,CY:? "
       {4 SPACES}":POSITION
       11,CY:? ASC(BYTE$(CY+
       1)):POSITION CX,CY
EL 330 A=USR(PMMOVE,2,ADR(BY
       TE$),24,170,128/PMMOO
       E)
BD 340 IF BTRIG(0)=0 THEN 340
BH 350 GOTO 150                ©
```

The New Automatic Proofreader For Commodore

Philip I Nelson, Assistant Editor

Now it's easier than ever to type in Commodore programs published in COMPUTE!. *This completely new version of the Automatic Proofreader is a significant improvement over the old Proofreader and catches almost any typing mistake that can be made. A single version now works on the Commodore 64, 128, VIC-20, Plus/4, and 16. Starting with this issue, all BASIC programs published in* COMPUTE! *for these computers are listed in the new Proofreader format. They cannot be checked with the old Proofreader.*

"The New Automatic Proofreader" is a short error-checking program that helps you type in COMPUTE! program listings without typing mistakes. The Proofreader conceals itself in memory and doesn't interfere with the program you're typing. Each time you press RETURN to enter a program line, the Proofreader displays a two-character value called a *checksum* in reverse video at the top of your screen. If you've typed the line correctly, the checksum on the screen matches the one in the printed listing—it's that simple. You don't have to use the Automatic Proofreader to enter COMPUTE!'s printed listings, but doing so greatly reduces your chances of making a typo.

Getting Started

First, type in the Automatic Proofreader program below *exactly* as it appears in the listing. Since the Proofreader can't check itself before it exists, type carefully to avoid mistakes. Don't omit any lines, even if they contain unfamiliar commands or you think they don't apply to your computer.

When you're finished, save at least two copies on disk or tape *before running it for the first time.* This is very important because the Proofreader erases the BASIC portion of itself when it runs, leaving only the machine language (ML) portion in memory.

When that's done, type RUN and press RETURN. After announcing which computer it's running on, the Proofreader installs the ML routine in memory, displays the message PROOFREADER ACTIVE, erases the BASIC portion of itself, and ends. If you type LIST and press RETURN, you'll see that no BASIC program remains in memory. The computer is ready for you to type in a new program listing.

Entering Programs

Once the Proofreader is active, you can begin typing in a BASIC program as usual. Every time you finish typing a line and press RETURN, the Proofreader displays the two-letter checksum in the upper-left corner of the screen. Compare this checksum with the two-letter checksum printed next to the corresponding line in the COMPUTE! program listing. If the letters match, you can be pretty certain the line is typed correctly. Otherwise, check for a mistake and correct the line.

The Proofreader ignores space characters that aren't enclosed in quotation marks, so you can omit spaces (or add extra ones) between keywords and still see a matching checksum. For example, these two lines generate the same checksum:

```
10 PRINT"THIS IS BASIC"
10 PRINT      "THIS IS BASIC"
```

However, since spaces inside quotation marks are generally significant, the Proofreader pays attention to them. For instance, these two lines generate different checksums:

```
10 PRINT"THIS IS BASIC"
10 PRINT"THIS ISBA      SIC"
```

A common typing mistake is transposition—typing two successive characters in the wrong order, like PIRNT instead of PRINT or 64378 instead of 64738. The old Commodore Proofreader couldn't detect transposition errors. Because the new Proofreader computes the checksum with a more sophisticated formula, it is sensitive to the *position* of each character within the line and thus catches transposition errors.

The Proofreader does not accept keyword abbreviations (for example, typing ? instead of PRINT). If you prefer to use abbreviations, you can still check the line with the Proofreader. Simply LIST the line after typing it, move the cursor back onto the line, and press RETURN. LISTing the line substitutes the full keyword for the abbreviation and allows the Proofreader to work properly. The same technique works for rechecking a program you've already typed in: Reload the program, LIST several lines on the screen, and press RETURN over them.

If you are using the Proofreader on the Commodore Plus/4, 16, or 128 (in 128 mode), *do not perform any GRAPHIC commands while the Proofreader is active.* When you perform a command like GRAPHIC 1, the computer moves everything at the start of BASIC program space—including the Proofreader—to another memory area, causing the Proofreader to crash. The same

thing happens if you run any program that contains a GRAPHIC command. The Proofreader deallocates any graphic areas before installing itself in memory, but you are responsible for seeing that the computer remains in this configuration.

Though the Proofreader doesn't interfere with other BASIC operations, it's always a good idea to disable it before running any other program. Some programs may need the space occupied by the Proofreader's ML routine, or may create other memory conflicts. However, the Proofreader is purposely made difficult to dislodge: It's not affected by tape or disk operations, or by pressing RUN/STOP-RESTORE. The simplest way to disable it is to turn the computer off, then on again.

A gentler method to disable the Proofreader is to SYS to the computer's built-in reset routine. Here are the SYS statements required for various Commodore computers:

Computer	Reset Command
64	SYS 64738
128	SYS 65341
VIC-20	SYS 64802
Plus/4	SYS 65526
16	SYS 65526

Inside The Commodore Proofreader

Writing a machine language program that works on five different computers is no small task. The first hurdle is finding a safe place to put the code. Though the cassette buffer is an obvious choice, it's located in different places on various machines, and putting ML there creates problems for tape users. Instead, the Proofreader uses 256 bytes of BASIC programming space.

Before it installs the routine in memory, the Proofreader checks which computer you're using. Then it stores the ML at the bottom of BASIC memory and protects itself by moving the computer's start-of-BASIC pointer to a spot 256 bytes higher in memory. Once that's done, the Proofreader activates the ML routine and erases itself with NEW. Note that because the Proofreader overwrites its first few BASIC lines, it's critical not to delete anything from the first portion of the program.

The ML portion of the Proofreader wedges into one of the operating system's built-in routines (CRUNCH). The system calls CRUNCH every time you enter a line from the keyboard (it can be a numbered program line or a direct command without a line number). Before the computer digests the line, it uses CRUNCH to convert BASIC keywords like PRINT into *tokens*—one- or two-byte numbers that represent the keyword. By changing the CRUNCH vector to point to the ML checksum routine, we can make the computer figure the checksum before it tokenizes the line with CRUNCH.

The checksum routine initially sets the checksum to equal the low-byte and high-byte values of the current line number. Then it scans the line, multiplying the ASCII value of each character by its position in the line and adding the result to the two-byte checksum as it moves down the line. After scanning the whole line, the Proofreader performs an *exclusive or* operation on the two bytes of the checksum and displays the final result as two alphabetic characters in reverse video. Though the final checksum could have been displayed as a two-digit hexadecimal number, the Proofreader uses letters so that no harm will be done if you accidentally press RETURN over the line containing the checksum.

Once this is done, the Proofreader restores everything to normal and jumps to CRUNCH, which handles the line as usual.

Commodore Compatibility

If you own a Commodore 64, you may already have wondered whether the Proofreader works with other programming utilities. The answer is generally yes, if you are using a 64 and if you activate the Proofreader after installing the other utility. There's no way to promise, of course, that the Proofreader will work with any and every combination of utilities you might want to use. Any program that disturbs the CRUNCH vector or the memory area where the Proofreader resides will probably crash the system without delay.

When using the Proofreader with another utility, you should disable both programs before running a BASIC program.

The New Automatic Proofreader For Commodore

```
10 VEC=PEEK(772)+256*PEEK(773)
   :LO=43:HI=44
20 PRINT "AUTOMATIC PROOFREADE
   R FOR ";:IF VEC=42364 THEN
   {SPACE}PRINT "C-64"
30 IF VEC=50556 THEN PRINT "VI
   C-20"
40 IF VEC=35158 THEN GRAPHIC C
   LR:PRINT "PLUS/4 & 16"
50 IF VEC=17165 THEN LO=45:HI=
   46:GRAPHIC CLR:PRINT"128"
60 SA=(PEEK(LO)+256*PEEK(HI))+
   6:ADR=SA
70 FOR J=0 TO 166:READ BYT:POK
   E ADR,BYT:ADR=ADR+1:CHK=CHK
   +BYT:NEXT
80 IF CHK<>20570 THEN PRINT "*
    ERROR* CHECK TYPING IN DATA
    STATEMENTS":END
90 FOR J=1 TO 5:READ RF,LF,HF:
   RS=SA+RF:H8=INT(RS/256):LB=
   RS-(256*H8)
100 CHK=CHK+RF+LF+HF:POKE SA+L
   F,LB:POKE SA+HF,H8:NEXT
110 IF CHK<>22054 THEN PRINT "
   *ERROR* RELOAD PROGRAM AND
   {SPACE}CHECK FINAL LINE":EN
   D
120 POKE SA+149,PEEK(772):POKE
    SA+150,PEEK(773)
130 IF VEC=17165 THEN POKE SA+
   14,22:POKE SA+18,23:POKESA+
   29,224:POKESA+139,224
140 PRINT CHR$(147);CHR$(17);"
   PROOFREADER ACTIVE":SYS SA
150 POKE HI,PEEK(HI)+1:POKE (P
   EEK(LO)+256*PEEK(HI))-1,0:N
   EW
160 DATA 120,169,73,141,4,3,16
   9,3,141,5,3
170 DATA 88,96,165,20,133,167,
   165,21,133,168,169
180 DATA 0,141,0,255,162,31,18
   1,199,157,227,3
190 DATA 202,16,248,169,19,32,
   210,255,169,18,32
200 DATA 210,255,160,0,132,180
   ,132,176,136,230,180
210 DATA 200,185,0,2,240,46,20
   1,34,208,8,72
220 DATA 165,176,73,255,133,17
   6,104,72,201,32,208
230 DATA 7,165,176,208,3,104,2
   08,226,104,166,180
240 DATA 24,165,167,121,0,2,13
   3,167,165,168,105
250 DATA 0,133,168,202,208,239
   ,240,202,165,167,69
260 DATA 168,72,41,15,168,185,
   211,3,32,210,255
270 DATA 104,74,74,74,74,168,1
   85,211,3,32,210
280 DATA 255,162,31,189,227,3,
   149,199,202,16,248
290 DATA 169,146,32,210,255,76
   ,86,137,65,66,67
300 DATA 68,69,70,71,72,74,75,
   77,80,81,82,83,88
310 DATA 13,2,7,167,31,32,151,
   116,117,151,128,129,167,136
   ,137
```

MultiMemory

For Commodore 64 And Apple

Patrick Parrish, Programming Supervisor

This short utility partitions free memory so several BASIC programs can be loaded into your computer at once. Among other things, it's a great aid during program development—you can keep a couple of BASIC programming utilities at hand as you work, or test alternate versions of new routines before adding them to your main program. The Apple version works on all Apple II series computers with either DOS 3.3 or ProDOS.

The idea of partitioning memory into several modules which can contain separate programs is not new—Charles Brannon relied on BASIC pointers to split the PET into four 8K blocks with "Quadra-Pet" (COMPUTE!, June 1981), and Feeman Ng later partitioned the 64 into three 12K blocks with "Triple 64" (COMPUTE!'s GAZETTE, April 1985). Much like these earlier programs, "MultiMemory" divides free memory in your Commodore 64 or Apple into independent workspaces. Three partitions are set up in the 64, and four in the Apple. As before, you can load different BASIC programs into the computer at once. These could be utilities, applications, or games. And, once again, you can save and load programs from any of these areas without affecting the others.

But MultiMemory goes one step further. Not only are the BASIC programs in each module protected from one another, but the variables generated by each are protected as well. Any program can change a variable's value without affecting identically named variables that may exist in other partitions. That means there's even less chance of conflict between the programs, allowing you more flexibility when using MultiMemory.

Entering MultiMemory

The BASIC languages in the 64 and Apple were both written by Microsoft, Inc. and thus share numerous similarities. Nowhere is this more clearly seen than in a section of memory called *zero page* (locations 0–255), where many BASIC pointers are stored. If you compare detailed memory maps for these computers, you'll find that many zero-page pointers, though located at slightly different addresses, are the same on these two machines. MultiMemory takes advantage of this by using identical zero-page pointers in the 64 (locations 43–56) and in the Apple (locations 103–116) to split up free memory. As a result, the 64 and Apple versions of MultiMemory are alike in many ways.

Whether you have a 64 or an Apple II series computer, MultiMemory is entered in the same fashion. Both versions contain short machine language routines entered by a BASIC loader. Carefully type Program 1 for the 64 or Program 2 for the Apple and save a copy to disk or tape before running it for the first time.

When you run MultiMemory, line 100 sets the top of BASIC memory for the first partition. This is memory location 16384 (64*256) on the 64 and location 8192 on the Apple (we'll see why later).

Line 110 POKEs the machine language routine in lines 150–330 into a safe place in memory. On the 64, it's placed at the top of BASIC RAM (40769) just below BASIC ROM (40960). On the Apple, it resides at location 38251 just below DOS (38400). These areas are relatively safe from memory conflicts.

Line 120 checks to see if the machine language data has been correctly typed. Lines 130 and 140 place zeros in three sequential locations at the start of each memory module. The first zero is required to indicate the start of BASIC. The second and third zeros NEW each memory module. Finally, the NEW command in line 140 clears the BASIC loader from memory and leaves us in module 1.

Three Or Four Computers

To access any module on the 64, type SYS 40769 and press RETURN. The cursor will disappear. Now, pick a work area by typing 1, 2, or 3. For a test, let's choose partition number 2. A tone sounds as the partition number is displayed on the screen. That's all it takes to switch modules.

If you're using an Apple, type CALL 38251 and press RETURN. Choose among work areas 1 to 4. Again, for a test, specify number 2. A tone sounds, the partition number is displayed, and we're ready to program.

When you type LIST, you'll see that module 2 is empty. Now type PRINT FRE(0) to determine how much memory is available in this

module. The 64 should show about 16K free, and the Apple about 8K. On either machine, there is plenty of room for a short BASIC program and its variables. To see that both a program and its variables remain intact within a particular module, let's enter and run a program in module 2 and then do the same in module 1.

While in module 2, type and run this program:

```
10 REM EXAMPLE 2
20 A$="MODULE #2"
30 FOR I=1 TO 20:NEXT I
```

After running this program, type PRINT A$,I and press RETURN. You should see this:

```
MODULE #2      21
```

Save this program to disk or tape with the filename "P2."

Now let's go to another module. Before we do this, list the program in module 2 so that it's at the bottom of the screen. Now type SYS 40769 on the 64 or CALL 38251 on the Apple and choose module 1.

Independent Variables

After entering module 1, type LIST to prove that our first program has been left behind in module 2. Again, if you wish, type PRINT FRE(0) to determine the amount memory available in this module. The 64 should show about 14K and the Apple approximately 6K.

Program "P2" should still be on the screen. Even though it was listed in module 1, it remains visible if you switch partitions without clearing the screen. This makes it possible to copy program lines from one module to another with the screen editor. Simply use the screen editing keys to cursor up to line 10. Change the 2 in this line to a 1 and hit RETURN to enter this line in memory (Apple users must cursor to the end of the line before pressing RETURN, of course). Line 10 is now copied into memory in module 1 *without disturbing line 10 in module 2.*

If the BASIC screen prompts do not obscure the other program lines, you can copy them to module 1 in the same manner. At any rate, lines 20 and 30 should read:

```
20 A$="MODULE #1"
30 FOR I=1 TO 10:NEXT I
```

As you did before, run the program and then type PRINT A$,I. The result is:

```
MODULE #1      11
```

Save this program on disk or tape as "P1."

Now, go back to module 2 with SYS 40769 or CALL 38251 and type LIST. You should see program "P2" on the screen unchanged. Print the values for A$ and I. You'll see they still have the values they had when we left module 2.

Applications

As you can imagine, this process can be very valuable if you're writing and debugging your own programs. Suppose you're writing a program in module 1 and you need a subroutine from a BASIC program you have on disk. Maybe you aren't sure which disk the program is on. On the 64, normally you'd have to save the program currently in memory before looking at the disk directory, because the directory overwrites the BASIC workspace. (On the Apple, this wouldn't be a problem, since the disk catalog is not loaded into the BASIC workspace.)

With MultiMemory, you can nimbly jump to another module, load the directory there to find the program you need, and then load it into that module or the third module, leaving the directory intact. And, once you've found the subroutine you need from your earlier program, you can list it on the screen, shift back to the first partition, and copy the lines by RETURNing over them.

Now suppose that some bug in your program keeps the subroutine from working as you expected. Since variables retain their values within each program module, you can test each routine separately, compare the resulting variables, and make changes to your working version where needed.

With all this jumping from module to module, you might loose track of which partition you're in. To find out, you can type PRINT PEEK(40914) on the 64 or PRINT PEEK(38339) on the Apple.

In addition to aiding program development, MultiMemory can be used to hold a few BASIC programs that can be run individually. For instance, you might have a series of BASIC programs that, in sequence, manipulate data stored on disk. The first program could read in the data, manipulate it, and then write the results back to disk. In turn, a second or third program (or on the Apple, even a fourth) stored in the other workspaces could do the same. Or you could designate one BASIC workspace for data storage, much like a RAM disk.

Before undertaking any sophisticated programming applications with MultiMemory, however, you should consider how it works and keep in mind a few restrictions on its use.

Memory Ceilings

As mentioned earlier, MultiMemory uses a series of BASIC pointers in zero page to set up each workspace. The values required by these pointers for each module are stored in a lookup table at the end of the program. Whenever you SYS or CALL MultiMemory, it stores the current zero-page values in positions within this table which correspond to the current module. The program then waits for you to specify the next module.

When you pick a module, MultiMemory reads the zero-page pointer values for the module and places them in their proper zero page locations. The pointers transferred are the starting addresses for the location of the BASIC program, the array and nonarray variables, the string variables, and the top of BASIC memory.

The barriers separating the partitions were selected to keep MultiMemory compatible with most programs. On the 64, the first module runs from memory locations 2048–16383; the second, from 16384–32767; and the third, from 32768–40768. Partitions are placed on 16K boundaries because the 64's VIC-II chip can address only one 16K block at a time. The VIC-II chip is responsible for handling the 64's video display—such things as sprite shapes, screen memory, and character data.

On the Apple, the first module runs from memory locations 2048–8191; the second, from 8192–16383; the third, from 16384–27317; and the fourth, from 27318–38250.

Note that the location of module 2 coincides with the Apple's first high-resolution graphics page. That means any programs which use the hi-res graphics page should be loaded into another module. And, of course, any programs loaded into module 2 will likely be erased if a program in another module uses the hi-res page.

Program 1: MultiMemory For Commodore 64

For instructions on entering this listing, please refer to "The New Automatic Proofreader for Commodore" in this issue of COMPUTE!.

```
XA 100  POKE56,64:CLR
CR 110  FORI=40769TO40959:READA
        :POKEI,A:X=X+A:NEXT
DH 120  IFX<>22456THENPRINT"ERR
        OR IN DATA STATEMENTS."
        :STOP
KA 130  POKE16384,0:POKE16385,0
        :POKE16386,0
RA 140  POKE32768,0:POKE32769,0
        :POKE32770,0:NEW
JD 150  DATA 174,210,159,189,21
        0,159,170,160,0,185
QJ 160  DATA 43,0,157,214,159,2
        32,200,192,14,208
SG 170  DATA 244,32,228,255,41,
        15,240,249,201,4
BA 180  DATA 176,245,72,72,169,
        35,32,210,255,104
QR 190  DATA 24,105,48,32,210,2
        55,169,13,32,210
BG 200  DATA 255,32,143,159,104
        ,170,142,210,159,189
CF 210  DATA 210,159,170,160,0,
        189,214,159,153,43
CR 220  DATA 0,232,200,192,14,2
        08,244,96,169,0
HS 230  DATA 168,153,0,212,200,
        192,25,144,248,169
PX 240  DATA 15,141,24,212,169,
        10,141,5,212,169
PK 250  DATA 84,141,15,212,169,
        39,141,1,212,169
MQ 260  DATA 20,141,4,212,169,2
        1,141,4,212,162
RA 270  DATA 0,134,251,160,0,20
        0,208,253,232,208
XD 280  DATA 250,230,251,165,25
        1,201,4,208,242,169
DD 290  DATA 0,141,4,212,96,1,0
        ,14,28,1
HJ 300  DATA 8,3,8,3,8,3,8,0,64
        ,0
JK 310  DATA 64,0,64,1,64,3,64,
        3,64,3
GE 320  DATA 64,0,128,0,128,0,1
        28,1,128,3
DR 330  DATA 128,3,128,3,128,65
        ,159,65,159,65,159
```

Program 2: MultiMemory For Apple

For instructions on entering this listing, please refer to "COMPUTE!'s Guide to Typing In Programs" in this issue of COMPUTE!

```
B6 100  HIMEM: 8192
B3 110  FOR I = 38251 TO 38399: R
        EAD A: POKE I,A:X = X + A
        : NEXT
6B 120  IF X < > 16759 THEN PRINT
        "ERROR IN DATA STATEMENT
        S.": STOP
5F 130  POKE 8192,0: POKE 8193,0:
         POKE 8194,0
B6 140  POKE 16384,0: POKE 16385,
        0: POKE 16386,0: POKE 273
        18,0: POKE 27319,0: POKE
        27320,0: NEW
B3 150  DATA 174,195,149,189,195,
        149,170,160
C6 160  DATA 0,185,103,0,157,200,
        149,232
2F 170  DATA 200,192,14,208,244,1
        73,0,192
52 180  DATA 201,128,144,249,141,
        16,192,170
A0 190  DATA 41,15,240,241,201,5,
        176,237
14 200  DATA 72,32,58,255,169,163
        ,32,237
C0 210  DATA 253,138,32,237,253,3
        2,142,253
00 220  DATA 104,170,142,195,149,
        189,195,149
31 230  DATA 170,160,0,189,200,14
        9,153,103
49 240  DATA 0,232,200,192,14,208
        ,244,165
C7 250  DATA 105,133,175,165,106,
        133,176,96
70 260  DATA 1,0,14,28,42,1,8,3
2F 270  DATA 8,3,8,3,8,0,32,0
47 280  DATA 32,0,32,1,32,3,32,3
6E 290  DATA 32,3,32,0,64,0,64,0
01 300  DATA 64,1,64,3,64,3,64,3
0E 310  DATA 64,182,106,182,106,1
        82,106,183
50 320  DATA 106,185,106,185,106,
        185,106,107
09 330  DATA 149,107,149,107,149
```

©

Experimenting With SID Sound

Mark A Currie

This versatile program for the Commodore 64 lets you experiment with a wide variety of sound effects and listen to how they sound in a preprogrammed song.

The Commodore 64's SID (Sound Interface Device) chip offers the ability to create a great number of rich, distinctive sounds. But sound programming involves so many different parameters (controlling values) that testing even a few different combinations can consume a lot of time. This program lets you change and experiment with any of the SID parameters quickly and easily. Once you have designed a sound, you can test its actual effect by playing it in a preprogrammed song.

Type in and save a copy of the program before running it for the first time. When you type RUN, the main menu displays 12 choices:

```
1 WAVEFORM              2 PULSE WIDTH
3 ATTACK/DECAY          4 SUSTAIN/RELEASE
5 LOUDNESS              6 SYNCHRONIZE V1 to V3
7 RING MODULATION       8 FILTERS
9 PLAY MUSIC           10 ZERO VARIABLES
11 QUIT                12 LIST OF EFFECTS
```

The first eight options let you design a sound by changing one or more of the SID chip's parameters. In each case, the computer tells you what range of values is allowed and prevents you from entering illegal values. If you select an option and decide not to change anything, press RETURN without entering any value. Let's look at each option in turn.

Option 1 lets you select one of four different SID waveforms—triangle, sawtooth, pulse, or noise. Each has its own distinctive tone. If you select the pulse waveform you must also set the pulse width to some nonzero value with Option 2. Options 3 and 4 permit you to adjust the ADSR (attack/decay/sustain/release) envelope of the sound. Each of the four ADSR parameters can be set separately to any number from 0–15.

Attack controls the rate at which the sound rises from zero to full volume. *Decay* controls the rate at which it falls from the level it reached at the end of the attack cycle to the volume level set by the sustain setting. *Sustain* sets the volume level the note will maintain from the end of the decay cycle until the note is turned off. And *release* controls how fast the note fades away after it's turned off. Since all four ADSR settings contribute to the ultimate result, it may take some experimenting to find exactly the envelope you want. Option 5 sets the SID's master volume control, accepting values from 0 (silence) to 15.

As you may know, the SID chip has three separate voices or tone generators. Option 6 permits you to synchronize two of these voices (1 and 3), producing effects which neither voice could produce alone. Note that voice 3's frequency must be set to some nonzero value before synchronization can work. This program lets you set voice 3's frequency to an unchanging value or to values that vary along with voice 1's frequency changes. Option 7 selects ring modulation: This effect is generated much like synchronization, but produces quite different effects. Again, you must set voice 3's frequency to a static value or an offset of the frequency for voice 1.

Note that while this program uses only voices 3 and 1 for synchronization and ring modulation, you may use these effects with other voice combinations in your own programs. In this case, voice 3's frequency affects voice 1. With the same techniques, you can affect voice 2 by voice 1's frequency, or affect voice 3 by voice 2's frequency. No matter which combination is used, you must supply frequency values for both of the voices involved.

Option 8 controls the filter, allowing you to change six different parameters. You may turn four different types of filters on or off and set the overall cutoff frequency and resonance for the filters. Filtering permits only certain specified frequencies to pass unchanged. Frequencies outside the specified range are much quieter or inaudible. You can choose from four types of filters: low-pass, band-pass, high-pass, and notch-reject. When a filter is on, you can also choose an overall filter resonance value.

Resonance emphasizes frequencies near the cutoff frequency of the filter.

Customized Music

When you select Option 9, the program plays a song using the sound you have designed. Don't be discouraged if the results aren't exactly what you expect at first. Sometimes a minor change in only one or two parameters (particularly those controlling the ADSR envelope) makes a big difference in the ultimate effect.

Option 10 clears all sound parameters to zero to clear the slate for a new sound. Note that the SID chip produces silence when every parameter is zero: The least you must do to produce a sound is turn on a waveform, set the volume to a nonzero level, and define some sort of ADSR envelope. To help you get started, this program begins by choosing maximum volume, a pulse waveform (with pulsewidth of 2048), an attack value of 1, and a decay of 9.

After designing a sound that you like, you may wish to use it in a program of your own. Option 12 generates two lists for that purpose. The first list summarizes the sound parameters currently in effect. The second list includes all the SID control locations and indicates which values to POKE into each register to reproduce the current sound. Although this program makes only one voice audible, you can achieve even more interesting effects by activating more than one voice at a time.

Sound Experimenter

For instructions on entering this listing, please refer to "The New Automatic Proofreader for Commodore" in this issue of COMPUTE!.

```
DA 100 POKE532B0,6:PRINT"{CLR}
       {7 DOWN}"SPC(9)"C64 SOU
       ND EXPERIMENTS":FORT=1T
       01300:NEXT
GG 110 L1=15:W1=65:A1=25:D1=9:
       S1=0:W2$="PULSE":P1=204
       8:L2=15:F1$="NO":P2%=8
JK 120 PRINT"{CLR}{DOWN}
       {5 SPACES}{RVS}
       {5 SPACES}SOUND EFFECT
       {SPACE}MENU{5 SPACES}
       {OFF}{4 DOWN}"
GE 130 PRINT" {RVS}1{OFF} WAVE
       FORM{9 SPACES}{RVS}2
       {OFF} PULSE WIDTH
       {DOWN}
KK 140 PRINT" {RVS}3{OFF} ATTA
       CK/DECAY{5 SPACES}{RVS}
       4{OFF} SUSTAIN/RELEASE
       {SPACE}{DOWN}
PD 150 PRINT" {RVS}5{OFF} LOUD
       NESS{9 SPACES}{RVS}6
       {OFF} SYNCHRONIZE V1TOV
       3
QC 160 PRINT" {RVS}7{OFF} RING
        MODULATION{2 SPACES}
       {RVS}B{OFF} FILTER/RESO
       NANCE{DOWN}
BK 170 PRINT" {RVS}9{OFF} MUSI
       C PLAYER{5 SPACES}{RVS}
       10{OFF} ZERO VARAIABLES
        {DOWN}
QX 180 PRINT" {RVS}11{OFF} QUI
       T{12 SPACES}{RVS}12
       {OFF} LIST OF EFFECTS
       {DOWN}"
KB 190 INPUT"{2 DOWN}ENTER NUM
       BER OF YOUR SELECTION";
       A
QA 200 ON A GOTO520,340,220,2B
       0,790,370,630,B20,1370,
       1030,1670,1040
FS 210 GOTO120
RQ 220 PRINT"{CLR}{2 DOWN}SET
       {SPACE}ATTACK VALUE BET
       WEEN 0 & 15"
BS 230 PRINT"{DOWN}CURRENT VAL
       UE OF ATTACK IS";(A1-D1
       )/16
PB 240 INPUT"{DOWN}";A3:A2=A3*
       16
SX 250 PRINT"{2 DOWN}SET DECAY
        VALUE BETWEEN 0 & 15"
XS 260 PRINT"{DOWN}CURRENT VAL
       UE OF DECAY IS";D1
HS 270 INPUT"{DOWN}";D1:A1=A2+
       D1:GOTO120
FC 280 PRINT"{CLR}{2 DOWN}SET
       {SPACE}SUSTAIN VALUE BE
       TWEEN 0 & 15"
QK 290 PRINT"{DOWN}CURRENT VAL
       UE OF SUSTAIN IS";(S1-R
       1)/16
DJ 300 INPUT"{DOWN}";S3:S2=S3*
       16
XX 310 PRINT"{2 DOWN}SET RELEA
       SE VALUE BETWEEN 0 & 15
       "
GE 320 PRINT"{DOWN}CURRENT VAL
       UE OF RELEASE IS";R1
BF 330 INPUT"{DOWN}";R1:S1=S2+
       R1:GOTO120
EE 340 PRINT"{CLR}{3 DOWN}SELE
       CT A PULSE WIDTH BETWEE
       N 0 & 4095"
HS 350 PRINT"{2 DOWN}CURRENT P
       ULSE WIDTH IS";P1
EX 360 INPUT"{DOWN}";P1:P2%=P1
       /256:P3%=P1-256*P2%:GOT
       O120
DA 370 PRINT"{CLR}{3 DOWN}DO Y
       OU WANT SYNCHRONIZING?"
FX 380 INPUT"TYPE Y OR N ";B$
FC 390 IFB$="N"THENW1=W1AND253
       :S2=0:GOTO120
JH 400 IFB$="Y"THENW1=(W1AND25
       1)OR2:GOTO420
GX 410 GOTO3B0
AE 420 PRINT"{2 DOWN}PRESENT F
       REQUENCY OF V3(VOICE3)I
       S";V4
PC 430 INPUT"{2 DOWN}ENTER FRE
       QUENCY FOR V3";V4:V3=IN
       T(V4*16.4)
QK 440 V4%=V3/256:V5%=V3-(256*
       V4%)
FD 450 PRINT"{2 DOWN}DO YOU WA
       NT V3 TO BE OFFSET
RG 460 PRINT"FROM FREQUENCY OF
        V1 BY"
SM 470 PRINT"AMOUNT YOU JUST E
       NTERED?"
XX 480 INPUT"{DOWN}TYPE Y OR N
       ";C$
BE 490 IFC$="N"THENS2=0:GOTO12
       0
CM 500 IFC$="Y"THENS2=1
JB 510 GOTO120
JP 520 REM SET UP WAVEFORM TYP
       E
GG 530 PRINT"{CLR}{2 DOWN}ENTE
       R THE FIRST LETTER OF T
       HE":PRINT"TYPE OF WAVEF
       ORM YOU WANT:
PP 540 PRINT"{DOWN}TRIANGLE, S
       AWTOOTH, PULSE, NOISE."
MX 550 PRINT"{DOWN}WAVEFORM IS
        NOW SET TO ";W2$
SE 560 INPUT"{DOWN}";W$
BF 570 IFW$="T"THENW1=17:W2$="
       TRIANGLE"
SM 580 IFW$="S"THENW1=33:W2$="
       SAWTOOTH"
RH 590 IFW$="P"THENW1=65:W2$="
       PULSE"
XC 600 IFW$="N"THENW1=129:W2$=
       "NOISE"
KD 610 IFW1=65GOTO340
HJ 620 GOTO120
FH 630 PRINT"{CLR}{2 DOWN}IF Y
       OU SELECT RING MODULATI
       ON THEN"
HA 640 PRINT"WAVEFORM WILL BE
       {SPACE}SET TO TRIANGLE
AQ 650 INPUT"{2 DOWN}DO YOU WA
       NT RING MOD? TYPE Y OR
       {SPACE}N";R$
FB 660 IFR$="N"THENW1=W1AND113
       :S2=0:GOTO120
QQ 670 IFR$="Y"THENW1=21:W2$="
       TRIANGLE":GOTO690
BB 680 GOTO630
KD 690 PRINT"{DOWN}FREQUENCY O
       F V3(VOICE 3) IS NOW SE
       T AT ";V4
SS 700 INPUT"{DOWN}SET FREQUEN
       CY OF V3 ";V4:V3=INT(V4
       *16.4)
XG 710 V4%=V3/256:V5%=V3-(256*
       V4%)
FE 720 PRINT"{2 DOWN}DO YOU WA
       NT FREQ. OF V3 TO"
RA 730 PRINT"TO BE OFFSET FROM
        V1 BY"
QE 740 PRINT"AMOUNT YOU JUST E
       NTERED"
CC 750 INPUT"{2 DOWN}TYPE Y OR
        N ";O$
BB 760 IF O$="N"THEN S2=0:GOTO
       120
KQ 770 IF O$="Y"THEN S2=1
FC 780 GOTO120
SR 790 PRINT"{CLR}{2 DOWN}LOUD
       NESS IS NOW SET AT";L1A
       ND15
RE B00 INPUT"{2 DOWN}SET LOUDN
       ESS BETWEEN 0 & 15";L2
PE B10 L1=L1AND240:L1=L2+L1:GO
       TO120
DS B20 PRINT"{CLR}{2 DOWN}ENTE
       R FIRST LETTER OF TYPE"
       :PRINT"OF FILTERING YOU
        WANT:
GJ B30 PRINT"{DOWN}LOWPASS, BA
       NDPASS, HIGHPASS,
XG 840 PRINT"OR NOTCH FILTER
       {DOWN}"
DC 850 PRINTF1$;" FILTERING IS
        BEING USED NOW.
FG 860 INPUT"{DOWN}TO TURN FIL
       TERING OFF ENTER 0";F$
```

```
HS 870 IFF$="Ø"THENL1=L1AND15:
       R2=Ø:F1$="NO":R6=Ø
QB 880 IFF$="N"THENL1=L2+80:R2
       =1:F1$="NOTCH"
PJ 890 IFF$="L"THEN L1=L2+16:R
       2=1:F1$="LOWPASS"
CG 900 IFF$="B"THEN L1=L2+32:R
       2=1:F1$="BANDPASS"
BC 910 IFF$="H"THEN L1=L2+64:R
       2=1:F1$="HIGHPASS"
QR 920 PRINT"{CLR}{2 DOWN}CUTO
       FF FREQUENCY IS NOW SET
        AT";F
MB 930 PRINT"{2 DOWN}INPUT CUT
       OFF FREQUENCY
AB 940 INPUT"BETWEEN Ø & 2047
       {SPACE}CYCLES";F
AG 950 F1%=F/8:F2%=F-(8*F1%):F
       $="?"
HE 960 PRINT"{2 DOWN}DO YOU WA
       NT TO SET RESONANCE
FF 970 INPUT"TYPE Y OR N";R$
BH 980 IFR$="Y"GOTO1ØØØ
FJ 990 R6=1:R5=Ø:GOTO12Ø
CH 1ØØØ PRINT"{DOWN}RESONANCE
       {SPACE}IS NOW SET AT";
       R5
KC 1Ø1Ø PRINT"{2 DOWN}SET RESO
       NANCE BETWEEN Ø & 15
KG 1Ø2Ø INPUTR5:R6=R5*16+1:GOT
       O12Ø
PR 1Ø3Ø CLR:F1$="NO":GOTO12Ø
ER 1Ø4Ø REM PRINT LIST OF SOUN
       D EFFECTS
JG 1Ø5Ø PRINT"{CLR}{DOWN}WAVEF
       ORM TYPE IS ";W2$
SX 1Ø6Ø IF W2$="PULSE"THEN PRI
       NT"PULSEWIDTH IS ";P1
JC 1Ø7Ø PRINT"{DOWN}ATTACK IS
       {SPACE}";(A1-D1)/16;"
       {2 SPACES}DECAY IS ";D
       1
KE 1Ø8Ø PRINT"{DOWN}SUSTAIN IS
        ";(S1-R1)/16;" RELEAS
       E IS ";R1
QP 1Ø9Ø PRINT"{DOWN}LOUDNESS I
       S SET TO ";L1AND15
AA 11ØØ IFR2=ØGOTO114Ø
HP 1111 PRINT"{DOWN}"F1$" FILT
       ER BEING USED WITH
EM 1120 PRINT "CUTOFF FREQUENC
       Y OF ";F
JJ 1130 PRINT"{DOWN}RESONANCE
       {SPACE}IS ";R5
MF 1140 IFW1AND2THENPRINT"
       {DOWN}SYNCHRONIZING IS
        ON":GOTO117Ø
CS 1150 IFW1AND4THENPRINT"
       {DOWN}RING MODULATION
       {SPACE}IS ON":GOTO117Ø
CQ 1160 GOTO119Ø
BH 1170 PRINT"POKE54286 WITH";
       V5%;"& POKE54287 WITH"
       ;V4%;
PP 11BØ PRINT"{4 SPACES}TO SET
        V3 FREQ TO";V4;"HZ
JK 1190 IFS2=1THENPRINT"FREQUE
       NCY OFFSET IS BEING US
       ED.
KH 12ØØ IFS2=1THENPRINT"SEE PR
       OGRAM LINES 1640-166Ø
GS 1210 INPUT"{DOWN}PRESS RETU
       RN TO CONTINUE";RB
EE 1220 PRINT"{CLR}{DOWN}VOICE
        1{3 RIGHT}VOICE 2
       {3 RIGHT}VOICE 3
       {3 RIGHT}VALUE TO
       {DOWN}"
JQ 1230 PRINT"REGISTER
       {2 RIGHT}REGISTER
       {2 RIGHT}REGISTER
       {2 RIGHT}BE POKED
       {DOWN}"
SC 1240 S(Ø)=P3%:S(1)=P2%:S(2)
       =W1:S(3)=A1:S(4)=S1
BQ 1250 FORS=ØTO4
KH 1260 PRINT 54274+S,54281+S,
       54288+S,S(S):NEXT
DQ 1270 PRINT"{2 DOWN}FILTERIN
       G & LOUDNESS"
EK 1280 PRINT 54293;TAB(30);F2
       %
MK 1290 PRINT 54294;TAB(30);F1
       %
QS 13ØØ PRINT 54296;TAB(30);L1
EP 1310 PRINT"{DOWN}RESONANCE
       {SPACE}& VOICE TO BE F
       ILTERED
HH 1320 PRINT"54295";TAB(30);
       {SPACE}R6AND24Ø
KD 1330 PRINT"ADD 1 TO 54295 F
       OR FILTERING VOICE 1
DG 1340 PRINT"ADD 2 TO 54295 F
       OR FILTERING VOICE 2
MR 1350 PRINT"ADD 4 TO 54295 F
       OR FILTERING VOICE 3
SA 1360 INPUT"PRESS RETURN TO
       {SPACE}CONTINUE ";R8:G
       OTO12Ø
SK 1370 RE=54272:RESTORE
GB 1380 FORR=54272 TO 54296: P
       OKER,Ø: NEXT
SD 1390 REM ENTER SOUND EFFECT
       S INTO SID
BD 14ØØ POKERE+2,P3%:POKERE+3,
       P2%
BB 1410 POKERE+5,A1:POKERE+6,S
       1:POKERE+14,V5%
JF 1420 POKERE+15,V4%:POKERE+2
       1,F2%:POKERE+22,F1%
KQ 1430 POKERE+23,R2:POKERE+24
       ,L1
CG 1440 READN,D:IFN=ØTHENPOKER
       E+24,Ø:GOTO12Ø:REM PLA
       Y MUSIC
DP 1450 H=INT(N/256):L=N-(256*
       H):DUR=16ØØ/D
AA 1460 POKERE+1,H:POKERE,L:PO
       KERE+4,W1
FC 1470 IF S2=1 GOTO163Ø
EM 1480 FOR T=1TODUR:NEXT
DR 1490 IFW1<1GOTO151Ø
JB 15ØØ POKERE+4,W1-1
SQ 1510 FORT=1TO2Ø:NEXT:GOTO14
       4Ø
JE 1520 DATA1ØB14,8,11457,8
DF 1530 DATA12860,4,12860,16,1
       4435,8,17167,8,10814,4
       ,11457,8,10814,8
RQ 1540 DATA9634,4,9634,16,114
       57,8,81Ø1,8,8583,4
DM 1550 DATA1Ø814,8,11457,8
MG 1560 DATA12860,4,12860,16,1
       4435,B,17167,8,10814,4
       ,11457,B,10814,8
GS 1570 DATA9634,4,9634,16,114
       57,8,81Ø1,8,8583,4
KB 1580 DATA12860,4,17167,16
QR 1590 DATA21269,4,21269,16,2
       1269,8,19269,8,17167,4
       ,12860,8,12860,8
HR 16ØØ DATA14435,4,17167,16,1
       7167,8,14435,8,12860,4
       ,1Ø814,4
GH 1610 DATA1Ø814,4,10814,16,1
       Ø814,8,9634,8,10814,4,
       12860,8,10814,8
SC 1620 DATA9634,4,9634,16,114
       57,8,8101,8,8583,4,Ø,Ø
MM 1630 REM DEVELOP OFFSET FRE
       QUENCY
KM 1640 F1=H*256+L:F1=F1-V3:IF
       F1<ØTHENF1=Ø
RA 1650 V4%=F1/256: V5%=F1-(25
       6*V4%)
QQ 1660 POKERE+14,V5%:POKERE+1
       5,V4%:GOTO148Ø
QK 1670 END                    ©
```

Mousify Your Applesoft Programs

Lee Swoboda

Here's the first installment of a two-part tutorial on interfacing and using a mouse with your Apple II computer. Though a mouse is preferred, you can use the techniques shown here to substitute a joystick or game paddles for the mouse.

Allow me to introduce a new word—*mousify*. Don't try to look that up in a dictionary—it's not there, at least not yet. Though Apple didn't invent the mouse, their Macintosh was the first popular home computer that used one; and it has changed the way that we interface with computers. So the need for the word *mousify*—to add mouse control to computer hardware or software—may grow.

A Mouse For Apple II

Apple II computers don't come with a mouse, but it's now possible to add one. If you have an Apple II, II+, or IIe, you'll need an interface card (Apple product A2M2050, $149.95 including the mouse). The interface is built into the IIc, so you only need to buy the mouse device (Apple product A2M4015, $99.95).

Attaching a mouse to your Apple II is only half the battle. To your computer, a mouse is merely another input device, like a joystick. Many new programs have the ability to accept mouse input. But there are thousands of existing programs that were written before the mouse appeared on the scene. Most commercial programs are written in machine language and protected from unauthorized access so, unless you are an expert machine language programmer, you won't be able to mousify them. However, Applesoft BASIC programs can easily be mousified.

Pages 35–40 of the *AppleMouse II User's Manual*, which comes with the mouse, give a brief description of how to use the mouse in a BASIC program. But the two examples are trivial and the text fails to mention some of the "features" that make mousifying an Applesoft program difficult. For example, you must use the INPUT statement to obtain mouse position parameters. Unfortunately, the INPUT command erases one line of screen text. In addition, your program must set the computer to receive input, which disconnects the keyboard. Mixing mouse and keyboard input requires switching input control between the mouse and the keyboard. Not only that, you have to use GET statements for keyboard input, which can be exasperating. Using these techniques quickly turns your BASIC program into a Rube Goldberg contraption of INPUTs, GETs and PRINT D$'s. But there's a better way to handle the mouse.

Fortunately, Apple's input and output (I/O) are *memory-mapped*, meaning that the keyboard and each character on the 40-column screen are at specific locations in Apple's main memory. Applesoft's PEEK and POKE commands let us examine and change characters on the screen without having to use INPUTs, GETs or PRINTs. This is especially important when using the mouse, since it lets you zoom quickly to any point on the screen to change a character, rather than perform complex string manipulations.

Practical Application

Let's see how this works. Type in and save Programs 1 and 2, then load and run Program 2, which creates a text file for Program 1 to read. When that's done, load and run Program 1. You'll see an input screen, typical of what might be used in an address book program that lets you store and recall names and addresses. (Of course, this program is just for demonstration; you can't use it to create a real address file.)

In a typical program, the computer would make you reenter an entire line to correct a misspelling or other error. Program 1 lets you point directly at an error with the mouse, and change only the incorrect character. The initial screen display should contain this information:

```
ENTER INFORMATION

FIRST NAME ... COMPUTE!
LAST NAME .... READER SERVICE
STREET ..... P.O. BOX 2141
CITY ..... RANDOR
STATE ..... PA
ZIP ..... 19089
TELEPHONE .... 1-800-334-0868

ERASE QUIT DONE HELP
```

Notice that the word READER is misspelled REEDER. That's the mistake you'll correct. Also notice there are three flashing rectangles

on the screen. The rapidly flickering rectangle in the upper-right corner is produced when Program 1 obtains mouse input (lines 10150–10160). This effect is always present, but is unimportant to the present discussion. The flashing C at the beginning of the word COMPUTE! is the cursor. The blinking reflex (^) in the upper left corner of the screen is the mouse pointer. The mouse moves the mouse pointer around the screen.

Move the mouse so the mouse pointer replaces the second E in REEDER and press the mouse button. The computer immediately moves the cursor to the same spot. Now type the letter A (upper or lower case) to correct the spelling mistake. That's all you need to do to correct the error. You can also use the arrow keys to move the cursor (Apple II uses the CTRL-J and CTRL-K keys to simulate the up and down arrow keys of the IIe and IIc). But the mouse moves the cursor much more rapidly, and is far more intuitive to use.

Now move the pointer to the word DONE in the strip menu at the bottom of the screen, and press the button. The computer reads the information from screen memory and, in this case, redisplays the updated information. Of course, in a working program you would replace lines 30120–30190 with routines that store the data for later recall. You can move the mouse pointer or cursor anywhere on the screen, but line 10710 of the program prevents you from typing anything outside the text area.

How The Program Works

Let's take a closer look at the significant portions of Program 1. Lines 130 and 10000–10830 are the most important. Line 130 sets the sensitivity of the mouse. When MI has a value of 20, the pointer moves to any part of the 40-column screen as the mouse moves within a 5 × 8 inch area. Give MI a larger value to make the mouse less responsive.

Lines 10070–10090 activate the mouse. Input control is transferred from the keyboard to the mouse, until line 20030 returns control to the keyboard. Lines 10150–10270 calculate the horizontal and vertical position of the mouse and determine whether the mouse button has been pushed. Line 10170 and lines 10440–10760 handle input from the keyboard. Note that input control remains with the mouse at this point: The program does *not* use the statement PRINT D$"IN#0" to return control to the keyboard. This is the key to the simplicity of Program 1, since it avoids the problems normally encountered when using GET and PRINT commands with DOS.

Line 10220 and lines 10230–10270 move the cursor to the same position as the mouse pointer when you press the mouse button. Lines 10320–10390 position the mouse pointer and return to line 10150 to read the mouse again. If you don't press a key or the mouse button, the computer stays in the loop from 10150 to 10390, reading the mouse and repositioning the mouse pointer. Lines 10590–10620 position the cursor. This routine is activated only when you press an arrow key or the mouse button. Lines 10640–10690 change all upper- and lower-case and all inverse characters to flashing.

Substituting A Joystick Or Game Paddles

If you don't have a mouse, you can use a joystick (or, less conveniently, game paddles) to achieve the same effects. With a few modifications, Program 1 can be made to accept joystick or paddle input. Here are the steps to follow:

1. Delete lines 120, 130, 10001–10090 and 20220.

2. Modify the following lines as shown:

```
C0 10150 X0=PDL(0)
84 10160 Y0=PDL(1)
A8 10170 IFB0>127THEN10440
A9 101B0 Y0=INT(Y0/10)+1
47 10200 X0=INT(X0/6)+1
68 10220 IFB0<128THEN10320
32 20030 REM
```

3. Add the following lines:

```
6D 10165 B0=PEEK(-163B4)
54 10215 B0=PEEK(-162B7)
```

After making these changes, resave Program 1, using a different filename to distinguish it from the original version. When you run it, the joystick moves the mouse pointer around the screen and the button works just like the mouse button. At this point you might wonder why anyone would buy a mouse, since a joystick or game paddle seems to work as a substitute. Part of the reason is simply preference—many people find that a real mouse "feels" better and is therefore more convenient than a joystick. More significantly, most commercial programs that accept mouse input do not recognize input from a joystick or paddles. If you're writing programs strictly for your own use, a joystick may serve the purpose; but if you buy commercial software that requires a mouse, you may have no choice.

Using a mouse is a new experience for many Apple II owners. I hope this program inspires you to mousify some of your own programs. In Part 2 of this article we'll expand the capabilities of Program 1 to let you use the mouse to delete and insert blocks of text.

Program 1. Apple II Mouse Demonstration

For instructions on entering this listing, please refer to "COMPUTE!'s Guide to Typing In Programs" published in this issue of COMPUTE!

```
BD 120 S0 = 2: REM SLOT CONTAINI
       NG MOUSE
D7 130 MI = 20: REM MOUSE SENSIT
       IVITY
5A 140 D$ = CHR$ (4)
BC 150 REM
B7 160 REM READ DATA FILE
98 170 REM
C8 1B0 PRINT D$"OPEN TEXT"
32 190 PRINT D$"READ TEXT"
6D 200 INPUT NF$,NL$,AD$,CI$,ST$
       ,ZI$,TE$
CD 210 PRINT D$"CLOSE TEXT"
B7 220 REM
25 230 REM DATA ENTRY SCREEN
BB 240 REM
4F 250 HOME
4D 260 Y1 = 4:X1 = 15:C0 = 160
35 270 INVERSE
D7 2B0 PRINT "                ENTER
       INFORMATION             "
2C 290 VTAB 24: PRINT " MENU:  E
       RASE  QUIT  DONE  HELP
            ";
C6 300 NORMAL
31 310 VTAB 4: HTAB 1
F4 320 PRINT "FIRST NAME .."
C6 330 PRINT "LAST NAME ..."
3C 340 PRINT "STREET ....."
D6 350 PRINT "CITY ......."
1F 360 PRINT "STATE ......."
36 370 PRINT "ZIP ........."
17 3B0 PRINT "TELEPHONE ..."
3A 390 VTAB 19: HTAB 10: INVERSE
     : PRINT "^";: NORMAL
81 400 PRINT " IS MOUSE POINTER"
3C 410 VTAB 21: HTAB 14: INVERSE
     : PRINT " ";: NORMAL
38 420 PRINT " IS CURSOR"
26 430 VTAB 4
5E 440 HTAB 15: PRINT NF$
```

```
66 450 HTAB 15: PRINT NL$
D9 460 HTAB 15: PRINT AO$
E1 470 HTAB 15: PRINT CI$
F6 4B0 HTAB 15: PRINT ST$
71 490 HTAB 15: PRINT ZI$
59 500 HTAB 15: PRINT TE$
73 9999 REM #10000
19 10000 REM
E6 10001 REM ---------------
29 10010 REM MOUSE ROUTINES
E6 10020 REM ---------------
39 10040 REM
A4 10050 REM TURN MOUSE "ON"
49 10060 REM
A8 10070 PRINT O$"PR#"S0: PRINT
      CHR$ (1)
CB 100B0 PRINT O$"PR#0"
B9 10090 PRINT O$"IN#"S0
17 10100 GOTO 10590
25 10110 REM
65 10120 REM DETERMINE POSITION
91 10130 REM OF MOUSE
3D 10140 REM
1C 10150 VTAB 1: HTAB 40
77 10160 INPUT "";X0,Y0,B0
7D 10170 IF B0 < 0 THEN 10440: R
      EM KEY PRESSED?
0B 101B0 Y0 = INT (Y0 / MI) + 1
7B 10190 IF Y0 > 24 THEN Y0 = 24
64 10200 X0 = INT (X0 / MI) + 1
75 10210 IF X0 > 40 THEN X0 = 40
BB 10220 IF B0 > 1 THEN 10320: R
      EM BUTTON PRESSED?
B9 10230 IF Y0 = 24 THEN 20030
63 10240 Y1 = Y0:X1 = X0
7B 10250 POKE V0,C0
4B 10260 C0 = C2
F2 10270 GOSUB 10800
F2 10280 GOTO 10620
69 10290 REM
ED 10300 REM POSITION MOUSE POIN
      TER
2D 10310 REM
B6 10320 IF V0 = V1 THEN C2 = C1
B0 10330 POKE V1,C2
A2 10340 V1 = 1023 + 128 * (Y0 -
      1) + X0
3F 10350 IF Y0 > B THEN V1 = V1
      - 984
9C 10360 IF Y0 > 16 THEN V1 = V1
      - 9B4
27 10370 C2 = PEEK (V1)
64 10380 POKE V1,160
BA 10390 IF C2 = 160 THEN POKE V
      1,30
C2 10400 GOTO 10150
31 10410 REM
B1 10420 REM KEYBOARD INPUT
41 10430 REM
F9 10440 C3 = PEEK ( - 16384)
77 10450 POKE - 16368,0
DC 10455 IF C3 > 223 THEN C3 = C
      3 - 32: REM CONVERT TO
      UPPER CASE
4B 10460 IF C3 > 159 THEN 10710
C0 10470 IF C3 = 141 THEN X1 = 1
      5:Y1 = Y1 + 1: IF Y1 >
      10 THEN Y1 = 4: REM RET
      URN KEY
69 10480 IF C3 = 139 THEN Y1 = Y
      1 + 1: REM DOWN ARROW
B2 10490 IF C3 = 138 THEN Y1 = Y
      1 - 1: REM UP ARROW
BF 10500 IF C3 = 149 THEN X1 = X
      1 + 1: REM RIGHT ARROW
71 10510 IF C3 = 136 THEN X1 = X
      1 - 1: REM LEFT ARROW
56 10520 IF Y1 > 24 THEN Y1 = 24
DC 10530 IF Y1 < 1 THEN Y1 = 1
9C 10540 IF X1 > 40 THEN X1 = 40
EB 10550 IF X1 < 1 THEN X1 = 1
5D 10560 REM
B6 10570 REM POSITION CURSOR
6D 10580 REM
A4 10590 POKE V0,C0
CA 10600 GOSUB 10B00
42 10610 C0 = PEEK (V0)
9E 10620 IF V0 = V1 THEN C0 = C2
44 10630 REM CHANGE TO FLASHING
      CHARACTER
B7 10640 C1 = C0
23 10650 IF C1 > 127 THEN C1 = C
      1 - 64
7F 10660 IF C1 > 64 THEN C1 = C1
      - 64
D9 10670 IF C1 > 95 THEN C1 = C1
      - 32
4B 10680 IF C1 < 64 THEN C1 = C1
      + 64
CB 10690 POKE V0,C1
CE 10700 GOTO 10150
6B 10710 IF X1 < 15 OR Y1 < 4 OR
      Y1 > 10 THEN 10150
DE 10720 GOSUB 10800
DC 10730 POKE V0,C3
51 10740 C0 = C3
CE 10750 IF V0 = V1 THEN C2 = C3
BC 10760 X1 = X1 + 1: IF X1 > 39
      THEN X1 = 39
67 10770 GOTO 10590
16 107B0 REM CALCULATE V0
6E 10790 REM (VIDEO BUFFER ADDRE
      SS)
6B 10800 V0 = 1023 + 128 * (Y1 -
      1) + X1
2B 10810 IF Y1 > 8 THEN V0 = V0
      - 9B4
7F 10820 IF Y1 > 16 THEN V0 = V0
      - 984
BB 10830 RETURN
9A 19999 REM #20000
1A 20000 REM
AE 20010 REM STRIP MENU
2A 20020 REM
C2 20030 PRINT O$"IN#0"
CB 20040 IF X0 > 8 AND X0 < 14 T
      HEN NF$ = "":NL$ = "":A
      O$ = "":CI$ = "":ST$ =
      "":ZI$ = "":TE$ = "": G
      OTO 250
1F 20050 IF X0 > 15 AND X0 < 20
      THEN HOME : END
73 20060 IF X0 > 21 AND X0 < 26
      THEN 30030
7A 20070 IF X0 > 27 AND X0 < 32
      THEN 20100
71 20080 VTAB 1: HTAB 40: PRINT
      O$"IN#"S0: GOTO 10150
17 20090 REM HELP TEXT
CD 20100 VTAB 12: HTAB 1
BA 20110 PRINT "THE FLASHING REF
      LEX (^) IS THE MOUSE"
75 20120 PRINT "POINTER AND THE
      FLASHING RECTANGLE IS"
4B 20130 PRINT "THE CURSOR.  TO
      MOVE THE CURSOR TO THE"
36 20140 PRINT "ENTRY YOU WANT T
      O CHANGE, USE THE ARROW
      "
4E 20150 PRINT "KEYS OR USE THE
      MOUSE TO MOVE THE MOUSE
      "
47 20160 PRINT "POINTER, THEN PR
      ESS THE MOUSE BUTTON TO
      "
E6 20170 PRINT "MOVE THE CURSOR
      TO THAT POINT.  TYPE"
EA 20180 PRINT "NEW OR CORRECTED
       DATA, THEN MOVE THE"
31 20190 PRINT "MOUSE CURSOR TO
      'DONE' IN THE MENU"
4A 20200 PRINT "BELOW AND PRESS
      THE MOUSE BUTTON TO"
D4 20210 PRINT "ACCEPT THE ENTRI
      ES ABOVE."
D9 20220 PRINT O$"IN#"S0
D3 20230 GOTO 10150
9D 29999 REM #30000
1B 30000 REM
2B 30010 REM EXAMPLE
2B 30020 REM
A1 30030 Y1 = 4: GOSUB 63050:NF$
      = A$
2C 30040 Y1 = 5: GOSUB 63050:NL$
      = A$
91 30050 Y1 = 6: GOSUB 63050:AO$
      = A$
1C 30060 Y1 = 7: GOSUB 63050:CI$
      = A$
E9 30070 Y1 = B: GOSUB 63050:ST$
      = A$
11 300B0 Y1 = 9: GOSUB 63050:ZI$
      = A$
17 30090 Y1 = 10: GOSUB 63050:TE
      $ = A$
2E 30100 REM GO TO REMAINDER OF
      YOUR PROGRAM
1C 30110 REM FOR EXAMPLE ...
36 30120 HOME
5E 30130 VTAB 10
EE 30140 PRINT NF$" "NL$
3B 30150 PRINT AO$
BB 30160 PRINT CI$", "ST$" "ZI$
9C 30170 PRINT TE$
CA 301B0 CALL - 19B: CALL - 19B
B9 30190 END : REM END OF EXAMPL
      E
A5 62999 REM #63000
24 63000 REM
2C 63010 REM SUBROUTINE TO "READ
      "
1F 63020 REM STRINGS FROM THE
B3 63030 REM VIDEO BUFFER
44 63040 REM
BF 63050 VTAB 24: FLASH : PRINT
      "  WORKING ...
                        ";: NO
      RMAL : VTAB 1: HTAB 1
C9 63060 A$ = ""
FC 63070 REM CALCULATE V0
55 63080 REM (VIDEO BUFFER ADDRE
      SS)
A5 63090 V0 = 1037 + 128 * (Y1 -
      1)
12 63100 IF Y1 > 8 THEN V0 = V0
      - 9B4
66 63110 IF Y1 > 16 THEN V0 = V0
      - 984
2F 63120 FOR I = 1 TO 25
67 63130 C0 = PEEK (V0 + I)
DD 63140 IF C0 = 160 AND PEEK (V
      0 + I + 1) = 160 THEN 6
      3190: REM END IF TWO BL
      ANKS
F9 63160 IF C0 > 128 THEN C0 = C
      0 - 128
F5 63170 A$ = A$ + CHR$ (C0)
D5 63180 NEXT I
C2 63190 IF RIGHT$ (A$,1) = CHR$
       (32) THEN A$ = LEFT$ (
      A$, LEN (A$) - 1): GOTO
       63190: REM REMOVE TRAI
      LING BLANKS
66 63200 RETURN
```

Program 2. Sample Screen Maker

```
51 10 O$ = CHR$ (4)
B7 20 PRINT O$"OPEN TEXT"
CF 30 PRINT O$"WRITE TEXT"
EA 40 PRINT "COMPUTE!"
BE 50 PRINT "READER SERVICE"
F1 60 PRINT "P.O. BOX 10958"
E3 70 PRINT "DES MOINES"
CB 80 PRINT "IA"
FC 90 PRINT "50950"
E9 100 PRINT "1-800-346-6767"
CC 110 PRINT O$"CLOSE TEXT"
```

Atari BootStuffer

Randy Boyd

This short, handy program for all eight-bit Atari computers lets you store as many as ten boot programs on a single disk and execute any of the programs just by pressing one key. A disk drive is required.

If you're like many Atari computer users, you probably have a number of disks that contain nothing but a single boot program. Even if you don't mind the expense of storing only one program per disk, that's not a very efficient arrangement. "Atari BootStuffer" allows you to put as many as ten boot programs on a single disk (depending on how long each program is), and still use each program as if it were alone on the disk.

Type in Atari Bootstuffer from the listing below, and save it. As listed here, the program works on an Atari 800 with an 810 disk drive. If you have an XL or XE model, change the numbers in line 750 as shown in the REM in line 740. If you have a 1050 disk drive with DOS 2.5 or 3.0 and wish to use enhanced-density, change lines 1140, 1170, 1300 and 1340 as indicated in the REM lines in the program listing. Changing those lines gives you 1040 sectors per disk (of course, this is not possible on an 810 disk drive, which doesn't support enhanced-density).

Creating A BootStuffer Disk

Before running Atari BootStuffer, format a disk to be used as the special BootStuffer disk. Now run the program and insert the freshly formatted disk in the drive. When you press the space bar, the screen turns green and the drive spins for about one minute. When the screen turns red, the special disk is ready to use. Reboot the system: The computer loads and executes a machine language program which lets you use the BootStuffer disk. When the prompt appears, you can press S to save a program on the disk or press L to load and run a program.

Since you just formatted the disk, it doesn't yet contain any programs you can load. Press S to choose the save option. The program indicates how many sectors are free in the current block and asks whether you want to load the target program from disk (press D) or cassette (press C). From that point onward, simply follow the prompts on the screen: The target program is loaded into memory and saved on the boot disk. If a load error occurs, the screen flashes red and the program starts over again. By repeating this process, you can save as many as ten boot programs on one disk (of course, the number of programs you can fit on one disk depends on how long they are).

BootStuffer prepares the disk by dividing it into ten blocks numbered 0–9, each containing 255 sectors. Since it uses the operating system boot routines, this program is not able to read sectors 256, 512, 768 or 1024. The BootStuffer code occupies the 13 lowest-numbered sectors on the disk, so a single-density disk can store programs only in sectors 13–255, 257–511 and 513–720. An enhanced-density disk with 1040 sectors can use all of the single-density sectors plus sectors 513–767, 769–1023 and 1025–1040.

It's important that you arrange the boot programs to fit into the Bootstuffer disk without wasting too many sectors. The program tells you how many sectors are left in the current block, and how many sectors are in the program you're trying to save. If a program is too large to fit in the current block, BootStuffer prompts you to save a smaller program in that block. If you don't have a smaller program, you can press N to advance to the next block. However, skipping to the next block wastes the free sectors remaining in the last block. If you try to save a program that requires more space than is left on the disk, BootStuffer generates a DISK FULL message and permits you to save a smaller program in the same space.

When you name a program to be saved on the disk, make sure the name is ten characters or less. Once you have saved as many programs as you want, put the BootStuffer disk in the drive and reboot the system, then press L to choose the load option. The contents of all ten blocks are displayed, and you're prompted to choose which program you want to execute. Press a number key from 0–9: The program in that block automatically loads from disk and executes. Blocks that don't contain a program are marked as empty. Don't select an empty block from the load menu: You may cause the system to crash.

Atari BootStuffer

For instructions on entering this listing, please refer to "COMPUTE!'s Guide to Typing In Programs" published in this issue of COMPUTE!

```
KL 500 FOR X=16384 TO 17920:
       POKE X,0:NEXT X
BA 510 ? "PLACE FORMATTED DI
       SK IN DRIVE"
EN 520 ? "PRESS SPACE BAR":?
       :? "■PLEASE WAIT■"
```

```
BH 530  IF PEEK(764)=255 THEN
        GDTD 530
OA 540  ST=1536:PDKE 710,192:
        PDKE 712,192
LO 550  READ J:IF J=-1 THEN G
        DTD 580
AI 560  PDKE ST,J:ST=ST+1
GP 570  GDTD 550
IH 580  ST=16384
LO 590  READ J:IF J=-2 THEN G
        DTD 620
AD 600  PDKE ST,J:ST=ST+1
GO 610  GDTD 590
EH 620  X=USR(1536)
LC 630  ? :? :PRINT "DDNE":PD
        KE 712,64:PDKE 710,64
CB 634  ? "PRESS SPACE BAR FO
        R ANDTHER CDPY"
LO 636  PDKE 764,255:IF PEEK(
        764)=255 THEN 636
DK 638  IF PEEK(764)=33 THEN
        GDTD 620
HB 640  END
CO 650  REM *** DISK SAVER **
        *******
JB 660  DATA 104,169,0,141,11
        ,3,133,240,141,4,3,16
        9,1,141,1,3,141
PA 670  DATA 10,3,169,49,141,
        0,3,169,87,141,2,3,16
        9,12,133,245,169
JD 680  DATA 64,133,241,165,2
        40,141,4,3,165,241,14
        1,5,3,24,165,240,105
EC 690  DATA 128,133,240,165,
        241,105,0,133,241,32,
        83,228,238,10,3,198,2
        45
DD 700  DATA 208,223,96,-1
JH 710  REM *** 8DOTSTUFFER *
        *******
LB 720  DATA 67,12,0,6,6,6,76
        ,34,6,162,0,160,0,169
        ,0,145,251
IE 730  DATA 200,192,0,208,24
        7,230,252,165,252,201
        ,15,208,237,76

IE 740  REM FOR XL/XE VERS.
        {4 SPACES}THIS IS 177
        ,197
BH 750  DATA 247,242
KH 760  DATA 0,169,35,133,251
        ,169,6,133,252,169,19
        6,141,200,2,141,198,2
        ,162
DJ 770  DATA 3,32,210,8,162,4
        ,32,210,8,32,117,8,16
        2,20,32,210,8
PE 780  DATA 32,117,8,32,5,9,
        32,245,8,201,76,208,1
        4,169,155,32,245
DL 790  DATA 8,32,143,9,32,7,
        7,76,9,6,201,83,208,4
        8,169,155,32
EH 800  DATA 245,8,32,117,8,3
        2,76,9,32,117,8,162,2
        1,32,210,8,32
CB 810  DATA 117,8,32,5,9,201
        ,67,208,6,32,64,10,76
        ,60,6,201,68
CH 820  DATA 208,9,32,189,6,3
        2,143,9,32,80,7,76,60
        ,6,162,11,32
KO 830  DATA 210,8,32,131,9,7
        6,60,6,162,15,32,210,
        8,162,16,32,210
LN 840  DATA 8,162,18,32,210,
        8,32,5,9,32,117,8,32,
        38,8,32,52
IG 850  DATA 8,96,32,252,6,32
        ,161,6,173,1,12,133,2
        53,32,149,9,32
AG 860  DATA 190,9,24,173,241
        ,11,109,1,12,144,27,1
        62,12,32,210,8,32
OG 870  DATA 5,9,201,89,240,1
        2,169,1,141,241,11,23
        8,242,11,32,76,9
HD 880  DATA 96,76,60,6,24,16
        2,19,32,210,8,32,117,
        8,96,169,49,141
LA 890  DATA 0,3,169,1,141,1,
        3,96,32,165,8,162,1,3
        2,210,8,32
LJ 900  DATA 5,9,72,32,245,8,
        32,117,8,104,24,233,4
        7,133,247,32,64
OB 910  DATA 9,169,0,105,12,1
        66,247,240,5,105,15,2
        02,208,251,170,189,91
AL 920  DATA 11,141,10,3,232,
        189,91,11,141,11,3,32
        ,117,8,96,162,255
FK 930  DATA 232,189,91,11,20
        1,197,208,248,134,248
        ,134,249,96,32,213,7,
        32
OJ 940  DATA 143,9,32,171,6,3
        2,77,8,174,1,12,134,2
        55,202,138,72,32
DI 950  DATA 65,8,104,170,224
        ,1,208,244,162,15,32,
        210,8,162,3,32,210
MH 960  DATA 8,162,18,32,210,
        8,32,5,9,32,117,8,173
        ,241,11,141,10
OF 970  DATA 3,206,10,3,173,2
        42,11,141,11,3,169,87
        ,32,52,8,166,255
FE 980  DATA 202,138,72,32,65
        ,8,104,170,224,1,208,
        244,32,95,8,32,176
BJ 990  DATA 7,162,10,32,210,
        8,96,169,0,141,4,3,16
        9,11,141,5,3
JM 1000 DATA 169,11,141,10,3
        ,169,0,141,11,3,169,
        87,141,2,3,32,83
FD 1010 DATA 228,48,4,32,65,
        8,96,76,150,6,32,65,
        7,32,117,8,162
AK 1020 DATA 6,32,210,8,169,
        11,133,245,32,5,9,32
        ,245,8,201,155,208
BA 1030 DATA 11,230,248,198,
        245,208,250,166,248,
        202,208,11,166,248,1
        57,91,11
AM 1040 DATA 230,248,198,245
        ,208,224,134,250,32,
        117,8,162,8,32,210,8
        ,32
OJ 1050 DATA 137,9,32,117,8,
        32,5,9,201,89,240,10
        ,32,117,8,165,249
ML 1060 DATA 133,248,76,216,
        7,96,32,252,6,169,0,
        141,10,3,141,11,3
LD 1070 DATA 169,82,96,141,2
        ,3,169,11,141,5,3,16
        9,128,141,4,3,32
IF 1080 DATA 123,8,32,83,228
        ,48,1,96,76,150,6,16
        6,250,232,24,173,241
AH 1090 DATA 11,157,91,11,23
        2,173,242,11,157,91,
        11,96,24,173,241,11,
        109
FA 1100 DATA 1,12,141,241,11
        ,144,8,173,242,11,10
        5,0,141,242,11,24,96
DB 1110 DATA 169,155,32,245,
        8,96,24,173,4,3,105,
        128,141,4,3,144,3
EK 1120 DATA 238,5,3,24,238,
        10,3,32,145,8,96,173
        ,10,3,201
JO 1130 REM FDR ENHANCED DEN
        SITY CHANGE LINE 114
        0 TO DATA 16
HK 1140 DATA 208
HN 1150 DATA 208,12,173,11,3
        ,201
HB 1160 REM FDR ENHANCED DEN
        SITY CHANGE LINE 117
        0 TO DATA 4
BF 1170 DATA 2
DG 1180 DATA 208,5,162,2,32,
        210,8,96,162,0,134
EH 1190 DATA 246,169,0,133,2
        45,138,72,189,91,11,
        32,245,8,104,170,232
        ,230
MH 1200 DATA 245,165,245,201
        ,13,208,237,138,72,3
        2,117,8,104,170,232,
        232,230
NF 1210 DATA 246,165,246,201
        ,10,208,216,96,232,1
        34,240,162,0,189,122
        ,10,232
AL 1220 DATA 201,155,208,248
        ,198,240,208,244,189
        ,122,10,232,134,254,
        32,245,8
AI 1230 DATA 201,155,240,4,1
        66,254,208,239,96,16
        2,11,142,66,3,162,0,
        142
AO 1240 DATA 72,3,142,73,3,7
        6,86,228,162,80,169,
        3,157,66,3,169,38
HK 1250 DATA 157,68,3,169,3,
        157,69,3,169,4,157,7
        4,3,169,0,157,75
JL 1260 DATA 3,32,86,228,169
        ,7,157,66,3,169,0,15
        7,72,3,157,73,3
CE 1270 DATA 32,86,228,133,2
        41,169,12,157,66,3,3
        2,86,228,165,241,96,
        201
KO 1280 DATA 10,176,5,201,0,
        144,1,96,76,60,6,173
        ,242,11,201
HA 1290 REM FOR ENHANCED DEN
        SITY CHANGE LINE 130
        0 TO DATA 4
BA 1300 DATA 2
AN 1310 DATA 240,24,56,169,0
        ,237,241,11,133,253,
        32,149,9,32,190,9,16
        2,7
KI 1320 DATA 32,210,8,162,0,
        76,210,8,56,169
KC 1330 REM FDR ENHANCED DEN
        SITY CHANGE LINE 134
        0 TO DATA 16
HH 1340 DATA 208
OK 1350 DATA 237,241,11,133,
        253,32
NH 1360 DATA 149,9,32,190,9,
        162,7,32,210,8,162,9
        ,76,210,8,169,64
EK 1370 DATA 141,198,2,96,16
        9,244,141,198,2,96,1
        69,196,141,198,2,96,
        169
AE 1380 DATA 48,133,225,133,
        226,133,227,56,165,2
        53,233,100,144,6,133
        ,253,230
KB 1390 DATA 225,16,246,56,1
        65,253,233,10,144,6,
        133,253,230,226,16,2
        43,230
```

```
IN 1400 DATA 227,198,253,208
        ,250,96,165,225,32,2
        45,8,165,226,32,245,
        8,165
BP 1410 DATA 227,32,245,8,16
        2,14,32,210,8,96,162
        ,16,169,3,157,66,3
AF 1420 DATA 169,4,157,74,3,
        169,128,157,75,3,169
        ,0,157,68,3,169,0
FB 1430 DATA 133,234,133,236
        ,169,6,157,69,3,169,
        12,133,235,32,86,228
        ,230
NJ 1440 DATA 236,169,7,157,6
        6,3,165,234,157,68,3
        ,165,235,157,69,3,16
        9
BK 1450 DATA 128,157,72,3,16
        9,0,157,73,3,24,165,
        234,105,128,133,234,
        165
NF 1460 DATA 235,105,0,133,2
        35,32,86,228,48,21,2
        06,1,12,208,206,169,
        12
PM 1470 DATA 157,66,3,32,86,
        228,48,6,165,236,141
        ,1,12,96,76,150,6
AD 1480 DATA 162,15,32,210,8
        ,162,13,32,210,8,162
        ,5,32,210,8,162,18
HH 1490 DATA 32,210,8,32,117
        ,8,32,5,9,169,12,141
        ,252,2,32,211,9
EO 1500 DATA 32,195,6,32,213
        ,7,32,143,9,32,252,6
        ,32,77,8,174,1
DN 1510 DATA 12,134,255,32,1
        09,7,96,155,84,72,73
        ,83,32,66,76,79,67
DJ 1520 DATA 75,155,83,69,76
        ,69,67,84,73,79,78,6
        3,155,198,213,204,20
        4
CD 1530 DATA 155,160,196,201
        ,211,203,173,203,210
        ,193,205,160,155,98,
        121,32,82
DP 1540 DATA 65,78,68,89,32,
        66,79,89,68,155,208,
        210,197,211,211,160,
        208
KG 1550 DATA 204,193,217,155
        ,78,65,77,69,32,63,1
        55,32,76,69,70,84,32
MD 1560 DATA 79,78,155,83,85
        ,82,69,63,32,89,47,7
        8,155,212,200,201,21
        1
NP 1570 DATA 160,196,201,211
        ,203,155,196,207,206
        ,197,155,194,207,207
        ,212,160,197
GG 1580 DATA 210,210,207,210
        ,155,211,205,193,204
        ,204,197,210,160,191
        ,155,66,79
CP 1590 DATA 79,84,32,84,65,
        80,69,155,45,83,69,6
        7,84,79,82,83,155
FN 1600 DATA 32,32,73,78,83,
        69,82,84,155,194,207
        ,207,212,173,173,196
        ,201
LA 1610 DATA 211,203,155,155
        ,200,201,212,173,173
        ,210,197,212,213,210
        ,206,155,208
LK 1620 DATA 210,207,199,210
        ,193,205,173,207,235
        ,161,155,76,41,79,65
        ,68,32
NM 1630 DATA 79,82,32,83,41,
        65,86,69,32,63,155,6
        7,41,65,83,83,32
BK 1640 DATA 79,82,32,68,41,
        73,83,75,32,63,155,4
        8,46,197,77,80,84
MG 1650 DATA 89,32,32,32,32,
        32,32,48,32,49,46,19
        7,77,80,84,89,32
ME 1660 DATA 32,32,32,32,32,
        49,32,50,46,197,77,8
        0,84,89,32,32,32
LO 1670 DATA 32,32,32,50,32,
        51,46,197,77,80,84,8
        9,32,32,32,32,32
MD 1680 DATA 32,51,32,52,46,
        197,77,80,84,89,32,3
        2,32,32,32,32,52
MJ 1690 DATA 32,53,46,197,77
        ,80,84,89,32,32,32,3
        2,32,32,53,32,54
AB 1700 DATA 46,197,77,80,84
        ,89,32,32,32,32,32,3
        2,54,32,55,46,197
MP 1710 DATA 77,80,84,89,32,
        32,32,32,32,32,55,32
        ,56,46,197,77,80
NJ 1720 DATA 84,89,32,32,32,
        32,32,32,56,32,57,46
        ,197,77,80,84,89
IL 1730 DATA 32,32,32,32,32,
        32,57,32,13,0,0,0,0,
        0,0,-2
```

Requester Windows In Amiga BASIC

Tom R. Halfhill, Editor

Here's how to add your own custom requester windows to any Amiga BASIC program. Like dialog boxes on the Macintosh, requester windows allow your programs to flag errors or request confirmation before carrying out important functions. The routine is written for Microsoft Amiga BASIC, which is now being shipped in place of MetaComCo ABasiC and is available as an upgrade to early Amiga owners.

Amiga BASIC is the most powerful BASIC interpreter supplied with any personal computer on the market. Written by Microsoft, it combines in a single language almost every feature found in IBM PC Advanced BASIC plus Microsoft BASIC for the Macintosh. In fact, many IBM BASICA and Macintosh BASIC programs will run on the Amiga with minor modifications.

However, Amiga BASIC does lack two key statements found in Macintosh BASIC: DIALOG and BUTTON. Both are important for writing BASIC programs which retain the mouse-and-window user interface common to the Macintosh and Amiga Workbench. Fortunately, both commands can be simulated fairly easily with Amiga BASIC's WINDOW and MOUSE statements.

In Macintosh BASIC, the DIALOG command lets a program open a *dialog box* (a small window) like those displayed by the Macintosh's operating system whenever the user must choose between two or more options. Dialog boxes also flag errors and alert users when they're about to activate a function that has irreversible consequences—such as quitting a program without saving the data on disk. For example, if the user pulls down a menu and selects Quit, a dialog box might open up and ask, "Quit program? (Data file not saved.)" Below this message is usually a pair of small boxes or circles called *buttons* which might be labeled OK and CANCEL. Pointing and clicking the mouse on the OK button exits the program; pointing and clicking on the CANCEL button cancels the Quit function and returns to the main program so the user can save his data if desired.

In Amiga BASIC, the DIALOG and BUTTON commands must be simulated by a routine that uses the WINDOW and MOUSE statements. For greater convenience, the routine can be written as a *subprogram*, another advanced feature included in Amiga BASIC. Subprograms are similar to subroutines, except they can have *local variables*. These are variables which are independent of the main program. For instance, if your main program uses a variable X for some purpose, a subprogram can also use a variable named X and it is treated as a separate variable. If the subprogram changes the value of its variable X, the main program's variable X is unaffected, and vice versa. On the other hand, a subprogram can also specify *shared variables*, sometimes known as *global* variables—those which are common to both the subprogram and the main program.

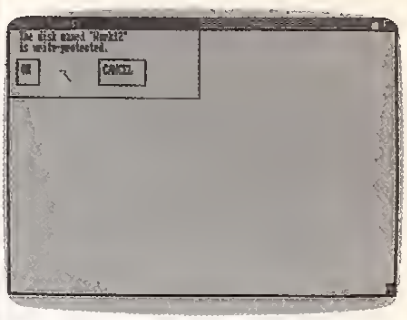
A major advantage of subprograms is that you can build up a library of useful routines on disk and add them to any new programs you write. This saves you the trouble of writing the same subprograms again and again. Although you can do the same thing with ordinary BASIC subroutines, there's always the chance that a subroutine variable might conflict with an identically named variable in your main program. Since subprogram variables are local, you're freed from this worry. Subprograms are truly programs within a program.

The Requester Subprogram

On the Amiga, dialog boxes are called *requesters*. Probably the most frequently encountered requester is the one that pops up when the Amiga asks you to insert a different disk. For the sake of consistency, an Amiga requester generally appears as a small window in the upper-left corner of the screen, has a title bar labeled System Request, has two or three buttons, does not have a resizing gadget or close gadget, and cannot be moved elsewhere on the screen.

The "Requester Window Subprogram" listed below duplicates most of these features. It creates a window that appears in the upper-left corner of the screen (or up to the full width of the screen in low-resolution modes); the window has a title bar labeled Program Request (to distinguish it from System Request windows); there is no resizing gadget or close gadget; and the window cannot be moved elsewhere on the screen. Unlike system requesters, this requester always displays two buttons, and they're always labeled OK and CANCEL.

The subprogram lets you display one or two lines of your own text in the Program Request window. The maximum number of characters allowed in each line depends on whether the Amiga has been set for 60- or 80-column text with the Preferences tool. If Preferences is set for 60 columns, each requester line can be up to 31 characters long. If Preferences is set for 80 columns, each line can be up to 39 characters. (You can adjust the subprogram for either mode by changing a single program statement; see the remarks in the listing.) If you try to display a line of text which exceeds these limits, the subprogram leaves off the extra characters. Since you won't know how Preferences is set if you're writing programs that might be used by other people, it's safest to assume 60 columns and restrict each line of your message to 31 characters.



A short subprogram lets you quickly and easily add custom requester windows to your own Amiga BASIC programs.

Requester Window Subprogram

```
RequeeterSub:
SUB Requeeter STATIC
 SHARED requeet1$,requeet2$,anewer:' Olobal variablee.
 ' Add ecreen parameter if needed to next line.
 WINDOW 2,"Program Requeet",(0,0)-(311,48),18
 ' Following linee truncate prompte if too long.
 ' If Preferencee ie eet for 60 columns,
 ' uee maxwidth=INT(WINDOW(2)/10) for next line;
 ' otherwiee uee maxwidth=INT(WINDOW(2)/8).
 maxwidth=INT(WINDOW(2)/10)
 requeet1$=LEFT$(requeet1$,maxwidth)
 requeet2$=LEFT$(requeet2$,maxwidth)
 PRINT requeet1$:PRINT requeet2$
 ' Thie eection drawe buttone.
 LINE (12,20)-(50,38),1,b
 LINE (152,20)-(228,38),1,b
 LOCATE 4,1:PRINT PTAB(20);"OK";
 PRINT PTAB(180);"CANCEL"
 ' Thie eection gete input.
 reqloop:
  WHILE MOUSE(0)=0:WEND:' Wait for button click.
  m1=MQUSE(1):m2=MOUSE(2)
  IF m1>12 AND m1<50 AND m2>20 AND m2<38 THEN
   anewer=1:' OK wae eelected.
   LINE (12,20)-(80,38),1,bf:' Flash OK box.
   WHILE MOUSE(0)<>0:WEND:' Wait for button releaee.
   WINDOW CLOSE 2:EXIT SUB
  ELSE
   IF m1>152 AND m1<228 AND m2>20 AND m2<38 THEN
    anewer=0:' CANCEL was eelected.
    LINE (152,20)-(228,38),1,bf:' Flaeh CANCEL box.
    WHILE MOUSE(0)<>0:WEND:' Wait for button releaee.
    WINDOW CLOSE 2:EXIT SUB
   ELSE
    GOTO reqloop
   END IF
  END IF
 GOTO reqloop
 END SUB
```

Opening a Program Request window is this simple:

```
request1$="This is the first line."
request2$="This is the second line."
CALL Requester
```

The two lines of your message are defined in the string variables *request1$* and *request2$*, and the CALL statement runs the subprogram (similar to GOSUB). The subprogram opens the requester window and waits for the user to click on the OK or CANCEL button. Clicks outside the buttons are ignored, although a click outside the requester window itself deselects it as the active window. It can be reselected, of course, by clicking within the window.

If the user clicks on OK, the subprogram returns a value of 1 in the variable *answer*. If the user clicks on CANCEL, *answer* equals 0. In either case, the subprogram closes the requester window after the button click and passes control back to the line following the CALL Requester statement. By testing *answer*, your program can branch to different routines to handle the user's response as required.

Hints For Use

Here's an example. Suppose your BASIC program sets up a Project

menu with a Quit selection (a consistent feature in Amiga software). When your MENU statement detects that Quit has been selected, it can GOSUB Quit:

```
Quit:
 MENU OFF:CLS
 request1$="Quit program?"
 request2$="(OK exits to Workbench or
   CLI.)"
 CALL Requester
 IF answer=0 THEN RETURN
 SYSTEM
```

If the user selects Quit by accident or changes his mind, he can click on CANCEL and no harm is done—the Quit routine merely RETURNs. Otherwise, a click on OK stops the program and exits BASIC with the SYSTEM command. Of course, you could also include a check to see if any data created with the program has been saved, and if necessary prompt the user to save it before quitting.

There are only two more details to keep in mind when using the requester routine. First, the WINDOW statement near the beginning of the subprogram opens WINDOW 2. If there's a chance that your program might already have two or more windows open when the requester is called, change this statement to WINDOW 3, or WINDOW 4, or whatever is necessary to avoid a conflict.

Second, the WINDOW statement defaults to the primary (Workbench) screen. That means the requester window always pops up on the primary screen. If your main program creates a secondary screen with the SCREEN statement, you'll want the requester window to appear on that screen instead of the primary screen. Otherwise, the requester will be invisible. To make the requester window appear on your program's secondary screen, append the screen's number to the WINDOW statement.

For instance, if your program creates a secondary screen with a statement such as this:

```
SCREEN 1,320,200,1,1
```

change the WINDOW statement in the requester subprogram as follows:

```
WINDOW 2,"Program Request",(0,0)-
  (311,45),16,1
```

This makes sure the requester will be visible. ©

Softkeys For Atari BASIC

Raymond Citak

This labor-saving utility adds automatic line numbering and 19 preprogrammed "soft" keys to your Atari computer. Even better, the soft key assignments are compatible with COMPUTE!*'s "Automatic Proofreader." For any Atari 400/800, XL, or XE computer with at least 48K RAM.*

If you write your own BASIC programs or enter the programs listed in COMPUTE!, you'll welcome any utility that cuts down on your typing time. "Softkeys For Atari BASIC" does exactly that—it gives you automatic line numbering and 19 preprogrammed soft keys that enter an entire BASIC word with just one keystroke. And there are two extra soft keys you can program for your own use.

Type in the program below and save it on disk or tape before running it for the first time. If you plan to use it along with the "Automatic Proofreader" to type in a COMPUTE! program, you should load and run Automatic Proofreader at this point. (Of course, this program works on its own, even if you're not using the Proofreader; but when the two are used together, you must install the Proofreader first.)

Now load and run the Softkeys program. It begins by asking you for the starting line number of the program you'll be typing in. Enter that number and press RETURN. Now you're asked to enter the increment (how much the line number increases between one line and the next). Most programs published in COMPUTE! are numbered in increments of ten, but you should always check the program listing to make sure. This number can be changed if the listing later changes to a different increment or skips some line numbers.

Automatic Line Numbering

After you enter this information, Softkeys installs its machine language portion in memory, deletes its BASIC portion, and leaves the computer ready for you to use. On the line below the READY prompt you'll see the first line number followed by a space. Type in the first line from the program listing, then

press RETURN to enter it in memory. The computer automatically prints the next line number and waits for you to enter the next line.

If the computer detects an error in the line, it prints the line again and shows where the error occurred. To retype the line, simply press SHIFT–DELETE, type in the correct line number and reenter the line. If you prefer, you can move the cursor back to the old line as usual, correct it, and press RETURN again. Just as in normal screen editing, the cursor can be anywhere on the line when you press RETURN.

The SHIFT–DELETE key combination also lets you perform several other tasks. If the program listing skips line numbers, press SHIFT–DELETE, then enter the new line number and continue typing as before. You can also use SHIFT–DELETE to enter any BASIC command from direct mode. For example, you may want to continue typing a program that you've partially entered and saved. Run Softkeys and answer the prompts as you did when you began typing the program. When the READY prompt comes back, press SHIFT–DELETE, then enter the command you would ordinarily use to load the program. After the program loads, the computer finds the last line number used in the program and automatically continues numbering from that point.

If you press SHIFT–DELETE and then change your mind, press RETURN without typing anything else: The correct line number reappears.

At times you'll need to change the line number increment midway through the program. To do this, press BREAK to disable Softkeys, then enter a USR statement in this format:

U=USR(39300,*line number,increment*)

The *increment* parameter specifies the desired new increment value, which takes effect *after* the next line is entered. The *line number* parameter must be included, but it has no effect. The program continues with the line number in use before the USR. For example, if you've been numbering lines by tens until you reach line 500 and wish to switch to increments of five, get a blank line by pressing SHIFT–DELETE when the computer prints 500, then enter the statement U=USR(39300,100,5). After this, the prompt for line 500 returns, but the next line number is 505.

Softkey Assignments

A *softkey* is a preprogrammed key combination that lets you print a complete BASIC command with a single keystroke. This program creates a number of softkeys that let you enter commonly used commands quickly and easily. The softkeys are all entered by pressing CTRL along with another key. The accompanying table lists all of the built-in softkeys.

When Softkeys is active, you can enter any of the 19 keywords shown in the table by pressing CTRL along with the indicated key. If you press CTRL-F, the computer prints FOR, and so on. This saves typing time and helps eliminate errors (the computer never types PRIMT instead of PRINT, for example). Note that STRIG and STICK both include a left parenthesis.

Atari Softkeys

Softkey	Command
CTRL-A	GRAPHICS
CTRL-C	COLOR
CTRL-D	DATA
CTRL-E	PEEK
CTRL-F	FOR
CTRL-G	GOTO
CTRL-I	INPUT
CTRL-K	STICK(
CTRL-L	LOCATE
CTRL-N	NEXT
CTRL-O	POKE
CTRL-P	POSITION
CTRL-R	READ
CTRL-S	SOUND
CTRL-T	STRIG(
CTRL-U	GOSUB
CTRL-W	DRAWTO
CTRL-?	PRINT
CTRL-RETURN	RETURN

Though 19 softkeys are built into the program, you can add two more of your own. To do this, you'll need to supply new values in the DATA statements in lines 1100 and 1180. Each line contains 10 values. The first value is the keyscan code generated when you press a key. Before you can program your own softkey, you need to know the keyscan code for the key combination you want to use.

For example, let's say you want to program the CTRL-V key combination to print the keyword SAVE followed by a quotation mark (SAVE"). To find the keyscan code for the CTRL-V combination (or any key combination), type the following statements in direct mode (without a line number) and press RETURN:

```
FOR J=1 TO 1E9:PRINT PEEK(764)
:NEXT J
```

The computer prints the keyscan code for whatever key or key combination is currently pressed. Try pressing different keys to see the numbers change. The number that appears when you press the desired combination is the keyscan code you need to use. In this case CTRL-V generates the keyscan value 144, so you should replace the first value in line 1100 with 144.

Encoding The Softkey

The next nine values in line 1100 represent the ATASCII values of the characters the computer should print when you press the designated softkey. The ATASCII values for the SAVE" character sequence are 83, 65, 86, 45, 34. Including the keyscan code, that comes to 6 values: Since you don't need the last four values in that line, make them all zeros (this DATA statement *must* have exactly ten values, even if you don't need to use all ten). When you're finished, line 1100 should look like this:

```
1100 DATA 144,83,65,86,45,34,0,0,0,0
```

To use your new softkey, simply rerun the program and try it out. By repeating the process, you can change line 1180 to add another, giving you a total of 21 softkeys. When programming a new softkey, note that you must include a space (character 32) if you want the cursor to move right one space after printing a keyword.

Occasionally, a program requires you to type in a character that requires a CTRL-key combination already used by Softkeys. Disable Softkeys by pressing BREAK, then enter the line. After that's done, you can reactivate the utility with a USR command as described above.

The machine language portion of this program resides in high memory just below the display list

in GRAPHICS 0. Use caution if you run a program and later issue the USR command to activate this utility. If the previous program used high memory for any purpose, the computer may crash.

Once you're satisfied with all the softkey assignments, you may want to convert the machine language portion of this program to a binary object file on disk. To do this, first run the BASIC portion, exit to DOS, select the Binary Save option, and save memory from $9984–$9BF1.

Softkeys For Atari BASIC

For instructions on entering this listing, please refer to "COMPUTE!'s Guide to Typing In Programs" in this issue of COMPUTE!

```
BI 10 DIM A$(3):? CHR$(125):
      ? :? "POKING DATA... P
      LEASE WAIT"
PE 20 FOR J=39300 TO 39921:R
      EAD A:POKE J,A:NEXT J
IM 30 ? CHR$(125)
CM 40 TRAP 40:POSITION 2,2:?
       "WHAT LINE NUMBER TO
      START WITH";:INPUT LN
LH 50 TRAP 50:POSITION 2,4:?
       "WHAT INCREMENT";:INP
      UT INC
KE 60 IF LN>=32767 OR INC>=3
      2767 THEN 30
LC 70 IF INC<=0 OR LN<0 THEN
      30
GP 80 TRAP 40000:? CHR$(125)
      :?
FO 90 IF PEEK(1614)=93 AND P
      EEK(1615)=6 THEN A$="A
      RE":GOTO 110
BJ 100 A$="IS":GOTO 120
MB 110 ? "THE AUTOMATIC PROD
       FREADER PROGRAM AND"
PA 120 ? "THE AUTONUMBER PRO
       GRAM WITH ";CHR$(34);
       "SOFT";CHR$(34)
MP 130 ? "KEYS ";A$;" NOW RE
       ADY FOR YOUR INPUT."
DI 140 ? "USE [redacted] TO DIS
       ABLE THE PROGRAM."
MG 150 ? "USE U=USR(39300,ln
       ,inc) TO ENABLE."
GD 160 U=USR(39300,LN,INC):N
       EW
BD 170 DATA 104,104,141,223,
       153,104
BE 180 DATA 141,222,153,104,
       141,221
BM 190 DATA 153,104,141,220,
       153,173
FJ 200 DATA 36,2,133,208,173
       ,37
FJ 210 DATA 2,133,209,169,5,
       133
FM 220 DATA 194,133,206,173,
       8,2
FJ 230 DATA 141,233,154,173,
       9,2
CI 240 DATA 141,234,154,169,
       193,141
AA 250 DATA 8,2,169,154,141,
       9
FM 260 DATA 2,174,6,228,232,
       142
MA 270 DATA 188,154,174,7,22
       8,142
AD 280 DATA 189,154,169,224,
       141,54
CD 290 DATA 2,169,153,141,55
       ,2
FH 300 DATA 160,3,162,154,16
       9,7
MF 310 DATA 32,92,228,96,0,0
FI 320 DATA 0,0,164,208,166,
       209
GK 330 DATA 169,7,32,92,228,
       173
FK 340 DATA 233,154,141,8,2,
       173
GC 350 DATA 234,154,141,9,2,
       169
CC 360 DATA 0,133,17,141,255
       ,2
II 370 DATA 141,240,2,133,77
       ,104
DJ 380 DATA 64,8,72,152,72,1
       38
GD 390 DATA 72,165,85,201,2,
       208
EM 400 DATA 25,173,242,2,201
       ,12
JM 410 DATA 208,18,169,23,22
       9,84
JE 420 DATA 48,12,165,194,20
       1,93
IJ 430 DATA 208,6,165,206,24
       0,11
CJ 440 DATA 198,206,104,170,
       104,168
JJ 450 DATA 104,40,76,98,228
       ,160
GA 460 DATA 1,177,136,16,13,
       173
BD 470 DATA 222,153,133,212,
       173,223
MB 480 DATA 153,133,213,24,1
       44,55
CM 490 DATA 165,136,133,204,
       165,137
LG 500 DATA 133,205,160,1,17
       7,204
MG 510 DATA 48,26,136,177,20
       4,133
BG 520 DATA 212,200,177,204,
       133,213
OD 530 DATA 200,177,204,24,1
       01,204
LF 540 DATA 133,204,165,205,
       105,0
PC 550 DATA 133,205,208,224,
       24,165
BJ 560 DATA 212,109,220,153,
       133,212
CD 570 DATA 165,213,109,221,
       153,133
PB 580 DATA 213,32,170,217,1
       65,212
LL 590 DATA 41,15,133,206,23
       0,206
IA 600 DATA 162,0,181,213,41
       ,240
PC 610 DATA 208,4,224,0,240,
       9
MP 620 DATA 74,74,74,74,9,48
LP 630 DATA 32,183,154,181,2
       13,41
DB 640 DATA 15,9,48,32,183,1
       54
CH 650 DATA 232,228,206,208,
       223,169
GH 660 DATA 32,32,183,154,16
       9,5
MI 670 DATA 133,194,133,206,
       76,40
MP 680 DATA 154,168,138,72,1
       52,32
MJ 690 DATA 0,0,104,170,96,8
GE 700 DATA 72,138,72,152,72
       ,44
QQ 710 DATA 9,210,16,22,162,
       0
JB 720 DATA 189,16,155,205,9
       ,210
HJ 730 DATA 240,21,160,0,232
       ,200
DD 740 DATA 192,11,208,250,2
       24,231
CI 750 DATA 208,236,104,168,
       104,170
PA 760 DATA 104,40,76,0,0,23
       2
GJ 770 DATA 189,16,155,240,6
       ,32
PH 780 DATA 183,154,24,144,2
       44,162
MB 790 DATA 126,142,31,208,1
       73,11
MP 800 DATA 212,205,11,212,2
       40,251
DJ 810 DATA 202,202,16,241,1
       04,168
LF 820 DATA 104,170,104,40,1
       04,64
FA 830 DATA 146,67,79,76,79,
       82
FG 840 DATA 32,0,0,0,0,186
HI 850 DATA 68,65,84,65,32,0
FO 860 DATA 0,0,0,0,189,71
KB 870 DATA 79,84,79,32,0,0
JD 880 DATA 0,0,0,128,76,79
MP 890 DATA 67,65,84,69,32,0
MD 900 DATA 0,0,138,80,79,83
BE 910 DATA 73,84,73,79,78,3
       2
BC 920 DATA 0,168,82,69,65,6
       8
DH 930 DATA 32,0,0,0,0,0
FC 940 DATA 190,83,79,85,78,
       68
FE 950 DATA 32,0,0,0,0,173
AM 960 DATA 83,84,82,73,71,4
       0
FG 970 DATA 0,0,0,0,141,73
MM 980 DATA 78,80,85,84,32,0
JK 990 DATA 0,0,0,136,80,79
IH 1000 DATA 75,69,32,0,0,0
PG 1010 DATA 0,0,170,80,69,6
        9
EJ 1020 DATA 75,40,0,0,0,0
DD 1030 DATA 0,163,78,69,88,
        84
BA 1040 DATA 32,0,0,0,0,0
HA 1050 DATA 166,80,82,73,78
        ,84
HJ 1060 DATA 32,0,0,0,0,133
DM 1070 DATA 83,84,73,67,75,
        40
CH 1080 DATA 0,0,0,0
MM 1090 REM CHANGE NEXT 10 B
        YTES TO INSERT YOUR
        DWN "SOFT" KEY.
AK 1100 DATA 173,83,84,82,73
        ,71,40,0,0,0
AB 1110 DATA 0,139,71,79
MD 1120 DATA 83,85,66,32,0,0
PK 1130 DATA 0,0,174,68,82,6
        5
MM 1140 DATA 87,84,79,32,0,0
DE 1150 DATA 0,140,82,69,84,
        85
IM 1160 DATA 82,78,32,0,0,0
MM 1170 REM CHANGE NEXT 10 B
        YTES TO INSERT YOUR
        DWN "SOFT" KEY.
DB 1180 DATA 168,82,69,65,68
        ,32,0,0,0,0
HM 1190 DATA 0,191
DB 1200 DATA 71,82,65,80,72,
        73
CL 1210 DATA 67,83,32,0,184,
        70
MC 1220 DATA 79,82,32,0
```

BASIC Sound On The Atari ST

Almost any music or sound effect can be created with the WAVE and SOUND commands in Atari ST BASIC. This article shows how to get started with ST sound and includes sample sound effects and a simulatea piano program. The article is an excerpt from the newly released COMPUTE!'s ST Programmers Guide *(by the editors of* COMPUTE!, *$16.95).*

The Atari 520ST contains a General Instruments sound chip that has three voices (sound channels) and a range of eight octaves. In fact, it's the same sound chip found in the MSX-standard computers sold in Japan and Europe. The chip's best feature is that it supports *envelope-oriented* sound—you can create a sound by defining the shape of its envelope. This allows considerable flexibility when designing sound effects and musical instrument tones. However, for programmers, it also requires more work than the SOUND command found in Atari BASIC for the eight-bit computers.

There are two sound commands in ST BASIC: WAVE and SOUND. WAVE controls the make-up of the sound:

WAVE *sound type, envelope, shape, period, delay*

Some of these parameters require values that toggle certain bits to activate certain functions. If you're not familiar with bit manipulation, refer to Figure 1. The first step is to decide which function(s) you want to select. Then add up the bit values—*not the bit numbers*—corresponding to those functions. The resulting number is what you use for that particular parameter in the WAVE statement.

For instance, the first parameter, *sound type*, controls whether a voice is set to noise, tone, or both. Bits 0–2, when set, turn on tone output for voices 1–3. Bits 3–5, when set, select noise output for the three voices. Both tone and noise may be turned on at the same time. Here are some example bit values:

WAVE 1—turns on tone for voice 1
WAVE 3—turns on tone for voice 1 and voice 2
WAVE 8—turns on noise for voice 1
WAVE 15—turns on tone and noise for all three voices

Bits 0–2 of *envelope* determine which of the three voices is controlled by the envelope generator. If a bit is set, its corresponding voice is controlled by the envelope generator.

The third parameter, *shape*,

Figure 1: Bit Values

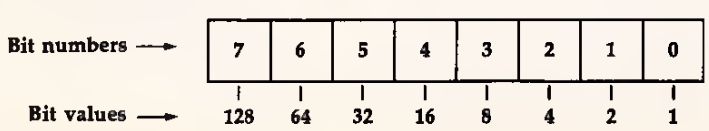
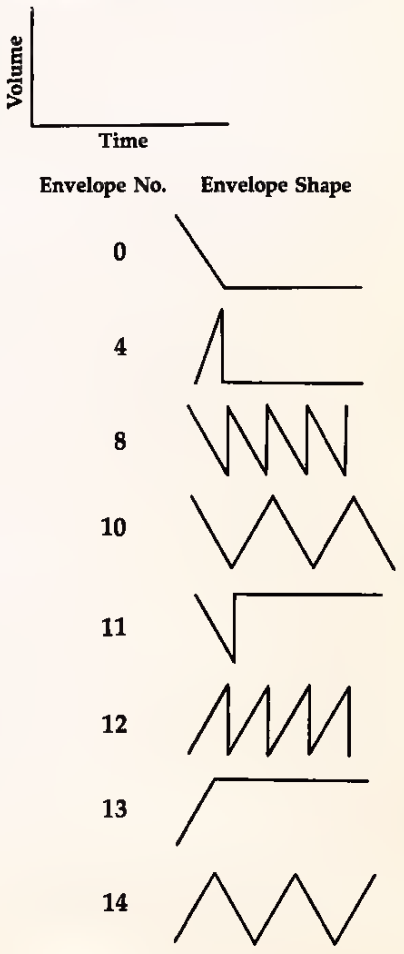


Figure 2: Available Envelopes



controls the way the sound's volume rises and falls. Figure 2 gives the possible values for this parameter and shows the shape of the subsequent sound.

Each of the envelope-shape drawings is a graph of volume over time. Take a close look at envelope zero and imagine what kind of sound it would make. The first thing to notice is that as soon as the sound begins, volume is at maximum. As time passes, the volume slowly fades away until it reaches zero. This type of sound is made by a piano. The hammer hits the string and almost immediately the volume reaches its maximum. The vibration of the string continues and slowly decays.

As you can see from Figure 2 envelopes 8, 10, 12, and 14 are repetitive. The sound continues to surge and fall long after the WAVE command is given.

The *period* parameter sets the period of the envelope, which is how fast the sound cycles. The larger the period, the longer the note takes to repeat. The last parameter, *delay*, controls the amount of time the program waits before executing the next BASIC command. Period is measured in one-fiftieth of a second increments. To hear a couple of interesting sound effects produced by WAVE, type in Program 1, "Helicopter," and Program 2, "Ding."

SOUNDing Off And On

The other music command, SOUND, turns on one of the voices for a specified duration. Its syntax is:

SOUND *voice, volume, note, octave, duration*

Voice selects which voice you want to turn on (1–3). *Volume* can be any number from 0 (off) to 15 (loudest). *Note* is a number from 1 to 12 and corresponds to the 12 notes in a scale (C, C#, D, D#, E, F, F#, G, G#, A, A#, B). The *octave* ranges from 1 (lowest) to 8 (highest). *Duration* can be any number from 0–65535. Each increment corresponds to one-fiftieth of a second.

Program 3, "Piano," uses the SOUND and WAVE commands to simulate a piano. Although it's intended as a sound demonstration, it also shows how to use other techniques, such as graphics and reading the mouse from BASIC.

You can run Program 3 in any graphics mode. When the piano keyboard appears on the screen, point to any key with the mouse and press the mouse button. The corresponding note is played.

Before typing in and running Program 3, you must make sure there's enough free memory available in BASIC. At this writing (mid-December), all 520STs were being shipped with the operating system (TOS) on disk. Later versions of the 520ST may be shipped with TOS in Read Only Memory (ROM). Until then, however, TOS must be loaded from disk into Random Access Memory (RAM). Because of the large amount of memory this requires, only a small area of storage remains for BASIC programs. When TOS and BASIC are loaded into a 520ST with 512K RAM, only about 5K is free for BASIC—enough for a program about 20 lines long. To check how much memory is available, load BASIC and type PRINT FRE(0).

Fortunately, there is a way to increase the amount of free memory by 32K. Normally, when windows are manipulated, the previous screen is saved in memory because part of it may be covered by a window and have to be restored later. The technique of saving the screen to memory is called *buffered graphics*. Although it can be quick and convenient, it requires 32K of memory to hold the screen.

If the buffered graphics option is turned off, 32K of memory is freed for BASIC. Click on Buf Graphics in the Run menu to toggle the buffered graphics on and off. This should increase free memory to 37986 bytes for BASIC programs.

Building The Piano

Let's trace through Program 3 to see how it works. Line 10 dimensions two arrays, B% and W%. These hold the note values of the black and white keys, respectively.

Next, the subroutine DRAWSCREEN is called. ST BASIC allows the use of labels instead of line numbers for many of its commands that need to make a reference to a line, like GOTO and GOSUB. Whenever you use a label in a line, make sure it is separated from the rest of the line with a colon.

The DRAWSCREEN subroutine (beginning at line 150) draws the piano keyboard. The first command of DRAWSCREEN sets the color of all screen output to black. Using only a single color for drawing ensures that the program will work in all graphics modes. The COLOR command also sets the fill pattern to solid. FULLW expands the window to full size, and CLEARW clears it.

The remaining commands of the DRAWSCREEN subroutine create the piano keyboard. Since there is no box drawing command in ST BASIC, we will simulate one using the LINEF command and FILL commands. LINEF draws a line between any two pairs of coordinates. The syntax is:

LINEF *xcoord1, ycoord1, xcoord2, ycoord2*

Next, line 20 calls the subroutine SETARRAY, which reads the note values of the black and white keys into the integer arrays B% and W%.

Reading The Mouse From BASIC

Now that the screen is set up and the arrays have been initialized, it's time to read the position of the mouse and check if the mouse button is pressed. This is done in the subroutine labeled READMOUSE at line 90. There is no BASIC command to read the mouse, so we must use one of the computer's Virtual Device Interface (VDI) routines. VDI routines are part of the computer's operating system.

The procedures necessary to call VDI routines are beyond the scope of this article, but basically involve POKEing various parameters into certain memory locations. These memory locations are not absolute addresses—instead, they're accessed via a reserved variable in ST BASIC named CONTRL. The ST automatically assigns an address to this variable which corresponds to the entry point into the VDI. By POKEing values into offsets from this address, various VDI routines can be executed.

The VDI routine for reading the position of the mouse and determining whether the mouse but-

ton is pressed has an opcode of 124, so we POKE CONTRL,124. We must tell the VDI routine that no other parameters are being passed, so two more POKEs are necessary: POKE CONTRL+2,0 and POKE CONTRL+6,0. Now we can call the VDI routine to read the mouse.

To read the horizontal and vertical position of the mouse, PEEK PTSOUT and PTSOUT+2, respectively. If the mouse button is pressed, PEEKing INTOUT will give a value of 1; otherwise, a zero is returned. (PTSOUT and INTOUT, like CONTRL, are also reserved variables for accessing VDI routines.)

The main loop of the piano program (line 30) simply waits until a mouse button is pressed. Once the button has been pressed, the vertical coordinates are checked to see if they are in the range of the piano keyboard (line 50). Then the vertical position is used to determine whether the key pressed is black or white (lines 50 and 60). If a black key is pressed, the note is calculated using the array B%; otherwise, the array W% is used.

Line 70 breaks the note value down into note and octave and then, using the SOUND command, plays the note.

Line 80 sets the envelope shape to zero. This creates a note with a similar shape to a piano's envelope. Program execution is then sent back to the main loop to check the mouse button again and SOUND another note when it is pressed.

Program 1: Helicopter

```
10 for a=1000 to 643 step -2
20 wave 8,3,14,a
30 for td=1 to 100:next:next
40 for a=643 to 1000 step 2
50 wave 8,3,14,a
60 for td=1 to 100:next:next
70 sound 1,0:sound 2,0
```

Program 2: Ding

```
10 for a= 1 to 12
15 sound 1,15,a,7
20 wave 1,1,14,5,1
30 for td=1 to 100:next:next
40 goto 10
```

Program 3: Piano

```
10 dim b%(16),w%(16)
20 gosub DRAWSCREEN:gosub
    SETARRAY
30 gosub READMOUSE:if button=0
    then 30
40 if y<70 or y>120 then 30
50 if y<100 then n=b%((x-16)/16.25)
60 if y>99 then n=w%((x-4)/16.25)
70 sound 1,15+15*(n=0),n-12*int((n-1)
    /12),3+int((n-1)/12)
80 wave 1,1,0,10000:goto 30
90 READMOUSE: poke contrl,124
100 poke contrl+2,0:poke contrl+6,0
110 vdisys(0)
120 x=peek(ptsout):y=peek(ptsout+2)
130 button=peek(intout)
140 return
150 DRAWSCREEN: color 1,1,1,1,1:fullw
    2:clearw 2
160 for a=50 to 100 step 50
170 linef 20,a,280,a:next
180 for a=20 to 280 step 16.25
190 linef a,50,a,100:next
200 for a=1 to 11:read s
210 gosub 250:next:return
220 data 32.5,48.75,81.25,97.5
230 data 113.75,146.25,162.5
240 data 195,211.25,227.5,260
250 linef s,50,s,78
260 linef s,78,s+8,78
270 linef s+8,78,s+8,50
280 fill s+1,51:fill s+5,51
290 return
300 SETARRAY: for a=1 to 16:read
    w%(a):next
310 for a=1 to 16:read b%(a):next
320 return
330 data 1,3,5,6,8,10,12,13
340 data 15,17,18,20,22,24
350 data 25,27
360 data 2,4,0,7,9,11,0,14,16
370 data 0,19,21,23,0,26,0
```


INSIGHT: Atari

Bill Wilkinson

Atari Character Codes

Last month's discussion about where and how to place things in memory served as a good lead-in to this month's topic: character codes. If you've read the heftier reference material, including COMPUTE! Book's *Mapping the Atari*, you may have discovered that your eight-bit Atari computer actually uses three different types of codes to represent the various characters (letters, numbers, punctuation, graphics symbols) it works with. All of these codes assign a unique number to represent each character, but the three codes are incompatible with each other because they use different numbering schemes.

The most commonly encountered code is called *ATASCII*, which stands for ATari-version American Standard Code for Information Interchange. Except for the so-called *control characters*—such as carriage return, tab, and so on—ATASCII is compatible with standard ASCII. (Why Atari chose to modify the standard is anyone's guess.) ATASCII is the character code used by PRINT, INPUT, CHR$(), ASC(), and most external devices such as printers and modems.

For example, in ATASCII (and ASCII), the code for uppercase A is 65. You can verify this in BASIC:

```
PRINT CHR$(65)
```

or

```
PRINT ASC("A")
```

Virtually every Atari BASIC book (even Atari's own) shows the character represented by each ATASCII code. You can also run Program 1 below to display each character and its code. (Press CTRL-1 to pause and continue the display.)

Screen Codes

The second character code found in your Atari is the *keyboard code*. The keyboard code for any character is actually the value read from a hardware register in memory when the key for that character on the keyboard is pressed. Program 2 below lets you find the keyboard code for any character. Just for fun, try some of the keys or key combinations which don't normally produce characters, such as CTRL-SHIFT-

CAPS). Neat, huh?

Finally: *screen codes*. This term refers to the byte value you must store in memory to display the desired character on the screen. "What?" you ask, "How do those differ from the ATASCII codes?" After all, to put the string BANANA PICKLE PUDDING on the screen, all it takes is a simple BASIC statement:

```
PRINT "BANANA PICKLE PUDDING"
```

And besides, aren't the characters in quotes supposed to be ATASCII codes? Good questions. Now for some complicated answers.

Actually, if the original Atari designers had thought just a little harder and added just a few more logic gates to the thousands already in the ANTIC and GTIA chips, ATASCII and screen codes could have been one and the same. It's similar to the mistake of making ATASCII incompatible with ASCII. Sigh. But we're stuck with what we've got, so let's figure out how it works.

For starters, consider GRAPHICS 1 and GRAPHICS 2, the large-size character modes. You may have noticed that in either of these modes you can display only 64 different characters on the screen. Now, if you recall last month's demo programs, note that we can specify the *base address* of the character set. That is, we can tell ANTIC where the character set starts by changing the contents of memory location 756 (which is actually a *shadow register* of the hardware location which does the work—see *Mapping the Atari* for more on this).

In a sense, the ANTIC chip is fairly simplistic. When it finds a byte in memory which is supposed to represent a character on the screen, it simply adds the value of that byte (multiplied times eight, because there are eight bytes in the displayable form of a character) to the character set base address. This points to the memory address for that particular character. Except...well, let's get to that in a moment.

Exception To The Rule

Because we want GRAPHICS 1 and 2 (with their limited sets of 64 different characters) to display numbers and uppercase letters (omitting lowercase letters and graphics), for these two modes it makes sense that the character set starts with the dot representation of the space character and ends with the underline—codes 32 through 95, respectively.

But why are these 64 characters the only ones available in GRAPHICS 1 or 2? Because the upper two bits of a screen byte in these modes are interpreted as *color* information, *not* as part of the character (see the modification to Program 3 below). So only the lower six bits choose a character from the character set. Six bits can represent only 64 possible combinations, which is why these modes can display only 64 characters. Bit pattern 000000 becomes a space, 100101 is an E, and 111111 becomes an underline, and so on.

When you use GRAPHICS 0 (normal text), however, there is a strange side effect. In this mode, only the single upper bit is the color bit (actually, it's the inverse video bit). This leaves 7 bits to represent a character, so we can have values from 0 to 127 decimal (0000000 to 1111111 binary, $00 to $7F hex). Again, this value—after being multiplied by eight—is added to the value of the character set base address. But which numbers in that 0 to 127 range represent which characters?

Well, we already know what the first 64 characters are—since the Atari's hardware limitations dictate that they *must* be the same as in modes 1 and 2. So the next 64 are the other characters. Program 3 illustrates how the ATASCII character set is linked to the screen set. Note how all the characters are presented twice, once in screen code (i.e., character ROM) order and once in ATASCII order. For some additional fun and info on modes 1 and 2, change line 10 to GRAPHICS 1. (Do not change it to GRAPHICS 2 unless you put a STOP in line 65 after the first FOR-NEXT loop.) Do you see what I mean about the upper two bits being color information?

Now you know why there are three different character codes used in your computer. How can you take advantage of this information? Well, if you combine this knowledge with the programs I presented last month, you could invent your own character set and design a word processor for some foreign language. (If you come up with a good Cyrillic character set, let me know.)

Actually, if you own an XL or XE machine, you have a second character set already built in. Just add this line to Program 3:

```
20 POKE 756,204
```

This tells the operating system and ANTIC that the base of the character set is at $CC00, which is where the international character set resides. Someday you might find some use for these characters. How will you know until you try?

For instructions on entering these listings, please refer to "COMPUTE!'s Guide to Typing In Programs" in this issue of COMPUTE!

Program 1: ATASCII Codes

```
OC 10 GRAPHICS 0
MK 20 FOR I=0 TO 255:PRINT I
      ,
ID 30 IF I=155 THEN PRINT "{
      RETURN}":GOTO 50
IC 40 PRINT CHR$(27);CHR$(I)
ON 50 NEXT I
NH 60 REM USE CONTROL-1 TO P
      AUSE
```

Program 2: Keyboard Codes

```
GC 10 DIM HEX$(16):HEX$="012
      3456789ABCDEF"
PK 20 POKE 764,255
LL 30 KEYCODE=PEEK(764)
GC 40 IF KEYCODE=255 THEN 30
MF 50 HI=INT(KEYCODE/16):LOW
      =KEYCODE-16*HI
LJ 60 PRINT "KEYCODE: HEX $";
OM 70 PRINT HEX$(HI+1,HI+1);
      HEX$(LOW+1,LOW+1);
JD 80 PRINT ", DECIMAL ";KEY
      CODE
AE 90 GOTO 20
```

Program 3: Screen Codes

```
OC 10 GRAPHICS 0
BP 30 SCREEN=PEEK(88)+256*PE
      EK(89)
BB 40 REM FIRST:  SCREEN COD
      E ORDER
EB 50 FOR C=0 TO 255:POKE SC
      REEN+C,C
OI 60 NEXT C
CO 70 REM THEN: {3 SPACES}ATA
      SCII ORDER
HA 80 SCR2=SCREEN+40*8
BH 90 FOR C=0 TO 255:CHAR=C
MP 100 IF C>127 THEN CHAR=C-
      127
EK 110 IF CHAR<32 THEN CHAR=
      C+64:GOTO 140
MB 120 IF CHAR>95 THEN CHAR=
      C:GOTO 140
ME 130 CHAR=C-32
JB 140 POKE SCR2+C,CHAR
BI 150 NEXT C
GH 999 GOTO 999:REM WAIT FOR
       BREAK KEY
```

The Beginners Page

Tom R. Halfhill, Editor

Cutting Strings Without Scissors

Now that we've covered the fundamentals of creating string variables over the past few columns, we can start exploring some of the more powerful string manipulations available in BASIC. Practically all BASIC languages have commands and functions for slicing strings of characters into smaller pieces, pasting two or more strings together to make longer strings, extracting certain sections from within strings, and inserting or replacing portions of strings. Some BASICs even have commands for rapidly searching through strings to find certain sequences of characters.

Since it may not be apparent why you'd want to do any of these things in your own programs, we'll show some common examples for each technique as we go along. In general, these functions give your programs the power to manipulate strings of characters for sorting, screen formatting, printing, storing and retrieving information, and other text-oriented operations.

Slicing Up Strings

Microsoft-style BASICs—such as those included with Commodore, Apple, IBM, Atari ST, and Amiga computers—generally have three functions for extracting shorter strings from longer strings: LEFT$, RIGHT$, and MID$ (pronounced "left-string," "right-string," and "mid-string"). TI BASIC has only one string function, SEG$, which is very similar to MID$. Atari BASIC, found on the 400/800, XL, and XE computers, handles string manipulations quite differently, as we'll see next month.

LEFT$ and RIGHT$ are easy to visualize: They extract characters from the leftmost and rightmost sections of a character string, respectively. You simply follow the keyword with the string variable you're extracting from and the number of characters you want to extract. For example:

```
10 A$="GEORGE WASHINGTON
   CARVER"
20 PRINT A$
30 B$=LEFT$(A$,6)
40 PRINT B$
50 B$=RIGHT$(A$,6)
60 PRINT B$
70 PRINT A$
```

When you type RUN, you should see this on the screen:

```
GEORGE WASHINGTON CARVER
GEORGE
CARVER
GEORGE WASHINGTON CARVER
```

To see how LEFT$ works, look at the statement B$=LEFT$(A$,6) in line 30. It grabs the leftmost six characters of A$—GEORGE—and stores them in B$. Line 40 confirms this by printing B$. To extract the phrase GEORGE WASHINGTON from A$, we could change line 30 to read B$=LEFT$(A$,17)—keeping in mind that spaces count as characters, just like letters, numbers, and symbols. (Of course, you can use your own variable names for A$ and B$ as long as you stick to this basic format.)

RIGHT$ is the opposite of LEFT$: It extracts the rightmost number of characters in A$ that you specify in the RIGHT$ statement. If you change line 50 to read B$=RIGHT$(A$,17), the result is WASHINGTON CARVER.

Line 70 shows that A$ remains intact after sections of it have been extracted with the LEFT$ and RIGHT$ functions. LEFT$ and RIGHT$ actually *copy* sections of the string into B$, rather than cutting the sections out.

Putting Lefty To Work

If you specify a value in a LEFT$ or RIGHT$ statement that is greater than the length of the string—in this case, say, B$=LEFT$(A$,35)—most BASICs return all of A$ in B$, the equivalent of B$=A$. This can happen in a program when you're unsure about the current length of A$, or if you're using a variable for the number parameter in a LEFT$ or RIGHT$ statement and the variable somehow is increased beyond the length of A$. If you specify a zero for this number—as in B$=RIGHT$(A$,0)—most BASICs return a null (empty) string.

If the number you specify in the LEFT$ or RIGHT$ statement is greater than 255, you'll probably get an error. Most Microsoft BASICs don't allow strings longer than 255 characters, so any reference to numbers greater than 255 in string-manipulation statements is invalid. Exceptions are the latest and most advanced Microsoft BASICs, such as Macintosh Microsoft BASIC and Amiga BASIC. They allow strings up to 32,767 characters long.

Of the two functions, LEFT$ is probably used more often than RIGHT$. One practical application of LEFT$ is to truncate user input to a predetermined length. For instance, let's say you're writing a program that asks for the user's name. At some point your program prints the name on the screen, but you want to limit the name to ten characters to keep from messing up your screen formatting. The solution is a line such as INPUT MYNAME$:MYNAME$=LEFT$(MYNAME$,10). Note that in this case, the original content of MYNAME$ *is* lost, because the LEFT$ function stores the leftmost ten characters back into MYNAME$.

Here's another application for LEFT$: Suppose your program asks the user a yes or no question. You can evaluate the answer with a line such as INPUT ANSWER$:IF LEFT$(ANSWER$,1)="Y" THEN GOTO 1000 (assuming that line 1000 is the beginning of your "Yes" routine). That way, your program responds correctly whether the user types Y, YES, YEAH, YEP, YES SIR, or even YOU BET. ©

Humanizing The User Interface, Part 1

Computers should be easy to use. Somehow this seems an obvious requirement for a product, yet many computer users are frustrated at the cumbersome nature of the programs they use day in and day out.

In previous columns, I've argued the case that computers should be transparent to their users—that the computer should disappear into the background, freeing the user to interact directly with the application. A key to transparent computing is the user interface—the vehicle through which the user interacts with the computer. The user interface has three components—input, output, and content.

Input generally involves the communication of physical motion from the user to the computer, signaling the computer to perform various activities. Typing on a keyboard, speaking into a microphone, or drawing a line with a finger on a touch tablet are all ways of using physical movement to convey information to a computer.

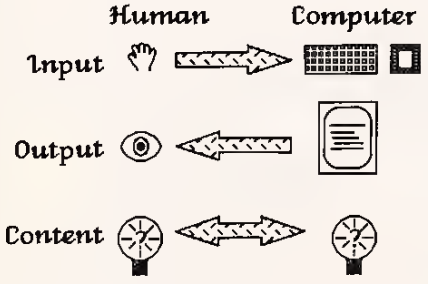
Output consists of messages communicated from the computer to the user's senses. The most often-used sense is vision—usually the screen display.

Content is the purpose of the computer activity—the management of text, the computation of spreadsheets, or the creation of graphic images, to name just a few. Although input flows from the user to the computer, and output goes from the computer to the user, the communication of content is purely inferential. In other words, the user has an internal model of what the computer program is doing, or how it is doing its task. To use a program successfully, it's not important if the user's model of what is happening is accurate. All that's important is if the model is consistent with the program's behavior.

Joy Or Pain

When we're working with a program that has a well-balanced user interface, computing is a joy. When the user interface is bad, we may think that computing just isn't worth the effort.

Fortunately there are a few good programs available that show how easy computers can be to use. Most users of *The Print Shop* (from Brøderbund) would agree that this product is wonderfully easy to use. Many people probably haven't read the instruction manual. This product also has good input and output interfaces that step the user through the creation of customized greeting cards, posters, banners, calendars, etc. This product is one of the top sellers of all time, so the role of a good user interface cannot be underestimated.



Quite often, software designers try to make their products easy to use by designing them to work with a modern input device like a mouse or touch tablet. Unfortunately, this isn't enough. For a computer application to appear transparent, the input, output, and content of the system must be meshed to create a combined ambience that is both natural to the user and appropriate to the task at hand. For example, any attempt to design input devices independently of the applications that use them is risky at best. A program that lets numbers be entered with a joystick may be appropriate for a game in which the joystick is used to select the number of players, but it is clearly the wrong approach for a financial analysis package that requires almost constant entry and update of numbers.

One reason I invented the KoalaPad was to make computers easier to use. Yet input devices like the KoalaPad are not enough by themselves. They can play an important role only when their use is a complementary part of the design of the whole product. This is why some people are frustrated by the Macintosh—not all Mac software is easy to use. It's true that this computer (and the Amiga) is capable of supporting tremendously powerful and easy-to-use software; but it's also true that many programs fall short in this important area.

It's hard to design a good user interface. Millions of dollars went into the research at Xerox that led to the desktop metaphor—the use of windows and pop-up menus that are now becoming commonplace. It took a heavy investment to bring the KoalaPad and Muppet Learning Keys to market. The cost of developing a good program for a personal computer can easily exceed $100,000. (Remember this the next time you think software costs too much!)

As difficult as this task may seem, those of us involved with computer software development owe it to our customers to make ease-of-use our top priority. The market slump of 1984 and 1985 showed that the public is unwilling to blindly accept everything thrown its way.

Next month we'll explore one model of human behavior that provides valuable clues in the search for the best user interfaces. ©

Fred D'Ignazio, Associate Editor

Snowflakes, Quilts, And Stained Glass Windows

Recently I reviewed the new Amiga from Commodore on public TV's *Educational Computing*. Afterward, I hoped to have a few days to play with the machine before returning it. But I hadn't reckoned with my kids.

They were hooked on the Amiga's mouse, windows, and brilliant colors the first time I turned on the computer. They played with it constantly. The only time I got on the machine was after their bedtime.

My children's favorite program was Electronic Arts' *Deluxe Paint*. It is the most spectacular microcomputer paint program I have ever seen. With its animated, cycling colors and its dozens of drawing and painting tools, it even surpasses the *MacPaint* program on the Macintosh. It is so seductive and so much fun to use that it qualifies as "computer popcorn" (see my column on "Computer Popcorn," COMPUTE!, January 1984). Once you start using it, it's almost impossible to stop.

Like my children, I quickly fell in love with the program. But I still had a nagging doubt. Computer popcorn is scrumptious. But is it also nutritious? How could my children and I use the program to feed our minds and imaginations? Could the program teach us to be artists?

Just A Doodler

So many computer art programs are of the easy-draw variety, like easy-cook microwave ovens and easy-play organs. They get you started drawing, playing, and having fun in no time at all. Then, *bonk!*, you bump into the limitations of your own skills, abilities, and imagination. You've become a super doodler, but you aren't any closer to making professional drawings, pictures, or art. That's because creativity programs, in general, are tools, not teachers. They are enormously enticing tools, but they can never replace a certain amount of training or skill.

Like many people whose artistic aspirations far exceed their abilities, I found this situation extremely frustrating. And I wondered how my children could acquire the skills to use this program properly. I couldn't teach them the skills, and neither could the program.

Then, suddenly, a solution appeared. One night my six-year-old son Eric was scribbling away on the Amiga with *Deluxe Paint*. "Do you like my picture?" he said, turning toward us. My wife and I looked up. We were astounded. From across the room, Eric's glowing picture resembled a stained-glass window. It could have adorned a medieval cathedral. It was beautiful!

Later, as I was falling asleep, I realized Eric had helped me stumble onto a way out of my dilemma. What we needed were images—images drawn from the real world and from works of art. We could study these images, copy them, and use them as inspiration to build new pictures of our own.

The Butterfly Maiden

The next day I went to the local library and checked out books on embroidery, quilting, needlepoint, and nature. The books were filled with images—colorful pictures of the diverse designs and patterns that man and nature can devise. These were to be our teachers.

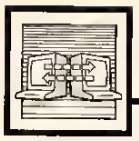
When I showed these images to my children, I concentrated on patterns and shapes that were symmetrical and geometric. Eric and my daughter Catie could draw these images effortlessly with the tools in *Deluxe Paint*. Catie especially liked the totem-pole faces on blankets woven by the Chilkat Indians of the Pacific Northwest; the brilliant colors and intricate geometric patterns found in nineteenth-century American pieced quilts; and pictures of the Butterfly Maiden, a Hopi kachina doll from northeastern Arizona.

I liked a tapestry, Nightsun, by the German artist Dirk Holger. Eric liked the Resurrection angels, saints, and serpents he found on stained-glass windows from South Africa, the French Loire, and Dublin, Ireland.

As we tried to copy these pictures, and those of Persian lions, helix-shelled snails, and the swirling atmosphere of Jupiter, we found that some images were easier to work with than others. Anything made with needlework, stitching, or embroidery was especially nice because the graph-paper patterns resembled pixels on the computer screen. Pure colors were easier than complex shadings and color blends. The blocky nature of many images was easy to reproduce on the computer, and big patterns made by endlessly repeating little patterns were easy to build using copy and paintbrush commands.

The next day, we went outdoors to look for images on our own. Our field trip turned up all sorts of new shapes: water spurting from the garden hose, wedding cakes at a local bakery, pine cones, and wildflowers. We carried many of these objects to the computer and tried copying their basic patterns. And at night we went back outdoors and looked up at the stars. When we grew cold, we came inside and drew dot-to-dot constellations.

We had found a solution to our problem. We had taken a first step toward becoming computer artists. And we did it by feeding our imagination fresh images, and by studying and copying these images to uncover their underlying patterns and designs. ©



Telecomputing Today

Arlan R Levitan

Games Modem People Play

When most people think about telecomputing, the first things that probably come to mind will be downloading public domain programs from electronic bulletin boards, retrieving stock quotes and financial information from commercial information services, or communicating with other hobbyists via Special Interest Groups (SIGs). Modems are often viewed as strictly utilitarian pieces of computer gear.

But there is a lighter side to telecomputing—multiple-player telegaming.

The first multiplayer telecomputing game I can recall involved a group of five or six people who were logged onto an online conferencing service playing *Dungeons and Dragons*. Players in California, Illinois, and New York were exploring the stygian depths of underground catacombs created by a Dungeon Master running the whole show from the keyboard of his Apple II in Austin, Texas.

The CompuServe Information Service was one of the pioneers in developing multiplayer online games. CompuServe currently offers a half-dozen or so such diversions to its subscribers. The blast-and-burn crowd can choose among multiple flavors of interstellar conflict: *SpaceWars, MegaWars I,* and *MegaWars III*. These games vary in both depth of play and the number of players who may simultaneously participate. *MegaWars III* is the clear heavyweight of the group. It has multiple game phases, including violent battle and economic warfare, and up to a hundred players can be pounding away at their keyboards at once.

Those with more pedestrian tastes may opt for a game of multiplayer blackjack, trading quips with the dealer and other players as electronic gambling chips trade hands.

Wheel Of Misfortune

Not all attempts at multiuser games are smash hits. CompuServe's latest creation is *You Guessed It!*, a TV-style quiz game in which players form teams and take turns attempting to answer questions while ignoring incredibly bad jokes delivered by an eerily obnoxious electronic master of ceremonies. The winners garner points that may be used to purchase gifts offered by sponsors, whose commercials regularly interrupt the game.

I tried *You Guessed It!* for about two hours, racking up what I thought was a respectable number of points. Then I eagerly issued the command that would transfer me to the "prize room" where players can trade points for their heart's desire. But the only prize I qualified for was a bumper sticker that advertised one of the *You Guessed It!* sponsors. To be fair, it did appear that if I played for another hour or two I could lay my hands on a baseball cap which sported (you guessed it) another advertiser's logo.

One of the more interesting experiments in telegaming that I've seen is a moderately obscure program called *COMM-BAT*, marketed by Adventure International. Some friends and I purchased copies of *COMM-BAT* for our Atari 800s back in early 1981 when 300 bits-per-second (bps) modems were still hot stuff for home use.

COMM-BAT lets two computers hook up over phone lines and presents each player with a battlefield map. The adversaries send tanks armed with rockets and lasers scurrying about in search of the enemy's base. When a player's base is destroyed, the game ends. The programs on both ends of the telecomputing link communicate with each other, updating the current battle information displayed on the screens. Players can also send insults and ultimatums to each other during the game.

A Reunion Battle

COMM-BAT does have its limitations. The character graphics are crude, but intentionally so. Versions of *COMM-BAT* were written for TRS-80, Apple, and Atari computers, and owners of these different systems could play *COMM-BAT* with each other and see identical displays on their screens. The biggest drawback was that the game progressed rather slowly due to the 300 bps modems.

Just for grins I pulled out my old copy of *COMM-BAT* and called one of my ex-buddies, now a resident of Denver, Colorado and a fellow user of GTE's PC Pursuit service (see "Telecomputing Today," December 1985). We cranked up our Ataris (now equipped with 1200 bps modems), linked up via PC Pursuit, and had a jolly old transcontinental time blasting the daylights out of each other. The extra speed of the 1200 bps modems and a noise-free connection transformed a mildly interesting game into good, clean Ramboesque fun. Out of curiosity, I called Adventure International and found that *COMM-BAT* is still available. The $49.95 price gets you all three versions of the program.

I'd like to hear about any other commercial or public domain telecomputing games that you may have encountered. I seem to recall some implementations of chess and *Battleship* having been done in the past. I'll compile a list and publish the results in a future issue.

Contact Levitan on The Source (TCT987), CompuServe (70675,463), or Delphi (ARLANL). ©

Donald B Trivette

The Ultimate Entertainment Center

Picture yourself in front of a 26-inch color monitor—shoes off, feet up, remote control in hand. But this is not just any remote control. This is a special remote unit that controls all of the components in your entertainment/computing system.

You push the TV button and bring up *World News Tonight* on the monitor: Peter Jennings reports that the stock market has soared to new highs. As he fades into a commercial, you decide to call Dow Jones News/Retrieval to see how your own stocks did. But first, you push the compact disc button to fill the room with a Beethoven symphony so real that you wonder where the orchestra is hiding. Then you press the VID2 key to put the computer video on the screen. You reach for the PCjr's wireless keyboard and start the appropriate communications program; then you press TV to return to the news while the computer retrieves the quotes.

At the next break, you display the Dow Jones results onscreen with the VID2 key. After the newscast, you press the VCR STOP, REWIND, and PLAY keys to view the "M*A*S*H" rerun you've been taping from an independent station. But first, you check the progress of the cassette tape you've been recording from an FM stereo broadcast.

This isn't a pipe dream—this is RCA's Dimensia. Billed as intelligent audio/video, it integrates numerous components into a single system commanded from a single remote control. The heart of the system is a 26-inch stereo monitor/receiver. Once you acquire the monitor, you can add other components according to your needs and budget. Current Dimensia components are an AM/FM receiver/amplifier, a compact audio disc player, a cassette tape recorder, two phonographs, a graphic equalizer, and several models of stereo VHS video recorders.

Connection Options

RCA designed the Dimensia system so you can also connect non-Dimensia components, including home computers. The PCjr, with its wireless keyboard, is a particularly good choice; it can be connected in three ways. Like most home computers and videogame machines, the PCjr can be hooked up to a TV's antenna terminals with an RF modulator. Since the Dimensia system allows multiple antennas—selected by remote control—you can switch between the PCjr's screen, cable service, and a satellite dish.

The PCjr also has a composite video output that can be connected to one of the monitor's three video input jacks. The PREVIOUS CHANNEL key lets you instantly switch between a TV program and the computer screen, so you can watch *Dynasty* and play *King's Quest* at the same time.

A third connection option is the Dimensia's RGB direct-drive video input. Although the Dimensia's RGB connectors aren't compatible with the PCjr's RGB plug, the signals *are* compatible. Radio Shack sells a four-conductor, color-coded patch cable that can be modified by anyone handy with a soldering iron to make the connection.

For everything but text, the Dimensia's composite video is as clear as the RGB mode, and it has an added advantage: You can record its output with a video cassette recorder. This means you can run programs on the PCjr and record the results on the VCR, which is perfect for putting titles on your home videos. You can also dub stereo audio from a compact disc player, the AM/FM tuner, the cassette recorder, or the phonograph.

A Piqued PCjr

Since both the Dimensia and the PCjr keyboard use an infrared remote control, there is the possibility of conflict. I couldn't find any button on the Dimensia's 52-key remote controller that the PCjr would recognize, but the computer was well aware that strange infrared signals were reaching its sensor. It squealed like a perturbed pig every time I used the Dimensia remote. This is easily and permanently solved by amputating Junior's little beeper—something I had intended to do for months anyway.

There's another annoying aspect of the PCjr you may want to fix, even if you don't have the Dimensia monitor. The joystick is not a wireless device and the cable that connects it to the computer is too short to reach across the room. Once again, it's Radio Shack to the rescue with its ten-foot joystick extension cord. Of course, this cord was designed for Tandy computers and the connections are not compatible with the PCjr's unusual plugs, so it's back to the soldering iron. Simply chop the joystick cable about eight inches from where it connects to the computer and solder a sub-D nine-pin connector (also available at Radio Shack) on each end, being careful to keep the pin numbers and wire colors consistent. It works perfectly.

The complete Dimensia system with all the components can cost as much as $5,000—but don't hesitate to haggle. The more components you buy, the better deal you can get.

Besides its flexibility, the Dimensia also may be the world's most user-friendly entertainment center. Although not documented in the manuals and unknown to sales people, the monitor displays a help screen across the bottom of the picture when you press AUX 0 0. Drop by a dealer and try it. ©

Programming the TI

C. Regena

IF-THEN Statements

IF-THEN statements are *conditional transfer* commands that make it seem as if computers can think. IF a specified condition is true, THEN the program skips to a certain line number elsewhere in the program; otherwise, the program simply continues to the next line as usual. TI BASIC also allows an ELSE statement as part of IF-THEN. It takes this form:

```
IF condition THEN line1 ELSE line2
```

IF the condition is true, THEN the computer goes to *line1*, or ELSE the computer goes to *line2*. If the optional ELSE is omitted, control merely passes to the following line. Here's a common example:

```
200 IF SCORE=10 THEN 900
210 PRINT SCORE
```

This statement says that if the value of the variable SCORE is equal to 10, then the program should continue at line 900. Otherwise, the program continues to the next line and prints the score.

You can use the other relational operators to define conditions in IF-THEN statements, too:

```
300 IF A<B THEN 700
400 IF X>Y THEN 200 ELSE 580
500 IF J<>8 THEN 800
```

In each case, the computer evaluates the condition—the expression between the words IF and THEN. If the expression is true, it has the value of −1. If the expression is false, it has the value of zero. Therefore, a statement such as this is valid:

```
320 IF A THEN 400
```

This doesn't look like the more common relational examples, but it implies that if A is equal to −1, then the program goes to line 400.

The condition may look more complex. If you keep in mind that true is −1 and false is zero, you can usually follow the logic. An example is:

```
150 IF (A=B)+C THEN 200
```

The part within the parentheses (A=B) is evaluated first. If A equals B, then the expression is −1 (true); if A does not equal B, the expression is zero (false). This value is then added to the value for C. If the result is −1, the condition is true and control passes to line 200.

Simulating AND/OR

Most other versions of BASIC allow the use of AND and OR in IF-THEN expressions. TI BASIC does not, but we can translate. Again, keep in mind that −1 indicates true.

Suppose we want to test the conditions A=B and C=D. If both are true (IF A=B AND C=D), then we want the program to continue at line 700. Here's one way to do this:

```
IF (A=B)+(C=D)=-2 THEN 700
```

If both conditions are true, each will yield −1 values, so the total will be −2.

Here's an equivalent way to make this test:

```
IF -(A=B)*(C=D) THEN 700
```

Notice that −1 times −1 is +1, so the negative sign in front converts the whole expression to −1 for true.

The word OR is used when one condition OR the other condition is true, but not both:

```
IF (X<Y) OR (X>Z) THEN 300
```

This can be translated to TI BASIC like this:

```
IF (X<Y)+(X>Z) THEN 300
```

Program control transfers to line 300 only if the expression evaluates to −1. This happens if only one of the conditions in parentheses is true (and thus −1) and the other is false (equal to zero).

Even more complex IF-THEN statements are possible by considering different combinations of + and * in evaluating conditions. Suppose after a CALL KEY statement the user may press either ENTER or any of the number keys. Here's the easiest way to set up the logic:

```
200 CALL KEY(0,K,S)
210 IF K=13 THEN 500
220 IF K<48 THEN 200
230 IF K>57 THEN 200
```

Or you can combine the IF statements like this:

```
210 IF (K<>13)+(K<48)+(K>57) THEN
    200
```

Algebra Drill

The sample program this month is a simple drill for beginning algebra students who are learning to add signed numbers. This program illustrates the use of several kinds of IF-THEN statements.

Lines 200 and 230 show two ways to check the length of the numbers to see if a randomly chosen number is negative. If necessary, a plus sign is added to the number.

Lines 280 and 300 determine the answer depending on the value of SUM.

If the answer is zero, line 360 skips the procedure for choosing the plus or minus sign in the answer. If the student needs to choose the sign, line 420 makes sure he or she presses either the plus sign or the minus sign. All other keys are ignored. Line 490 then receives the number keys pressed.

Line 530 checks the student's answer and branches appropriately. Line 590 waits for the student to press the ENTER key before continuing.

If you wish to save typing effort, you can obtain a copy of "Adding Signed Numbers" by sending a blank cassette or disk, a stamped, self-addressed mailer, and $3 to:

C. Regena
P.O. Box 1502
Cedar City, UT 84720

Adding Signed Numbers

```
100 REM  ADDING SIGNED NU
    MBERS
110 CALL CLEAR
```

```
120 PRINT "ADDING SIGNED
    NUMBERS":::
130 SCORE=0
140 FOR PROB=1 TO 10
150 T$=""
160 RANDOMIZE
170 A=INT(19*RND)-9
180 B=INT(19*RND)-9
190 A$=STR$(A)
200 IF LEN(A$)=2 THEN 220
210 A$="+"&A$
220 B$=STR$(B)
230 IF LEN(B$)>1 THEN 250
240 B$="+"&B$
250 PRINT "ADD"
260 SUM=A+B
270 S$=STR$(SUM)
280 IF SUM<>0 THEN 300
290 S$=" "&S$
300 IF SUM<=0 THEN 320
310 S$="+"&S$
320 TA=8-LEN(S$)
330 PRINT :TAB(4);A$
340 PRINT :TAB(4);B$
350 PRINT TAB(3);"---":::
360 IF SUM=0 THEN 450
370 CALL KEY(0,K,S)
380 CALL HCHAR(23,TA,45)
390 CALL HCHAR(23,TA,32)
400 CALL HCHAR(23,TA,43)
410 CALL HCHAR(23,TA,32)
420 IF (K<>43)+(K<>45)=-2
    THEN 370
430 CALL HCHAR(23,TA,K)
440 T$=CHR$(K)
450 FOR J=1 TO LEN(S$)-1
460 CALL KEY(0,K,S)
470 CALL HCHAR(23,TA+J,63
    )
480 CALL HCHAR(23,TA+J,32
    )
490 IF (K<48)+(K>57)THEN
    460
500 CALL HCHAR(23,TA+J,K)
510 T$=T$&CHR$(K)
520 NEXT J
530 IF SUM<>VAL(T$)THEN 5
    60
540 PRINT ::"CORRECT!"
550 SCORE=SCORE+1
560 PRINT :"THE SUM IS ";
    S$
570 PRINT ::"PRESS <ENTER
    >."
580 CALL KEY(0,K,S)
590 IF K<>13 THEN 580
600 CALL CLEAR
610 NEXT PROB
620 PRINT "OUT OF 10 PROB
    LEMS,"
630 PRINT :"YOUR SCORE IS
    ";SCORE:::
640 END                    ©
```

News & Products

Of Nordic Gods On The 64

Eurosoft International, a software publisher that specializes in introducing European software products to North America, has announced the release of *Valhalla*. Winner of the 1984 British Microcomputing Game of the Year Award, *Valhalla* is an animated, interactive game involving Nordic mythology. Thirty-six mythological characters are featured, each with a different personality. The player interacts with each of these in pursuit of the lost treasure of Valhalla. The mythological characters, shown using the "MoviSoft" animation technique, can either help or hinder your quest depending on their disposition and your actions.

Valhalla is available for the Commodore 64 at a list price of $24.95.

Eurosoft International, 114 East Ave., Norwalk, CT 06851

Circle Reader Service Number 200.

IBM PC MIDI Editor

MIDI Ensemble, a new software package from Sight & Sound for owners of musical equipment with a MIDI interface, consists of three main program modules: Recorder, Event Editor, and Phrase Editor. The Recorder module can be used for recording and overdubbing tracks; the Event Editor enables precise editing of pitch, start time, duration, and key-strike velocity; and the Phrase Editor allows copying, moving, deleting, combining and modifying musical phrases of any length. Also included is a text and graphics editor for creating diagrams or comments with a song file.

MIDI Ensemble runs on the IBM PC; list price, $495.

Sight & Sound Software, 3200 S. 166th St., New Berlin, WI 53151

Circle Reader Service Number 201.

Word Processor For Atari ST

Written by the developers of *AtariWriter* and *AtariWriter Plus*, *Regent Word* is a sophisticated, easy-to-use word processing program for the Atari ST. It features 80-column editing, function key-driven commands, local and global searches, multiple type fonts, print preview, and a communications package. It retails for $49.95.

Regent Spell is an expandable spelling checker for *Regent Word*. The program is shipped with 30,000 words; another 30,000 can be added. Misspelled words are highlighted in context. Commands can be issued via the ST's mouse or though single keystrokes. It also retails for $49.95.

Regent Software is also in the process of designing *Regent Base*, a database management program for small business use.

Regent Software, 7131 Owensmouth, #4SA, Canoga Park, CA 91303

Circle Reader Service Number 202.

Home Inventory Package For The 64

What's Our Worth?, from ADITA Enterprises, is a program designed to help you do a complete inventory of your personal belongings. Screen instructions and prompts make it very easy to enter items into inventory, read all items, search for specific information, change or delete items, and make a backup data disk.

What's Our Worth? is available by mail order, and retails for $19.95.

ADITA Enterprises, 116 Bermondsey Way N.W., Calgary, Alberta, Canada T3K 1V4.

Circle Reader Service Number 203.

Educational Enchantment

Sunburst has released *The Enchanted Forest*, a mathematics learning program with a fairy tale setting for grades four and up. The game begins when the witch of the forest transforms all of the forest animals into geometric shapes of different sizes and colors and hides them in ponds. Players travel through

the forest with 12 friends, using the concepts of conjunction, disjunction, and negation to break the witch's spells.

The Enchanted Forest was written by Dr. Jerzy Cwirko-Godycki, author of more than 40 children's books. It's available for the 64K Apple II+, IIe, and IIc; and the IBM PC and PCjr. The $59 list price includes a backup disk and teacher's guide.

Sunburst Communications, 39 Washington Avenue, Pleasantville, NY 10570.

Circle Reader Service Number 204.

Beach-Head Sequel For Atari

Beach-Head II, Access Software's sequel to the popular *Beach-Head* game, is now available in a version for the Atari 400/800, XL, and XE series with at least 48K of RAM. Like its predecessor, *Beach-Head II* is a World War II era arcade game that is set on the beaches of Europe. The sequel has several new features including voice synthesis, multiple play screens and play levels, sound effects, and animation techniques.

Beach-Head II for Atari lists for $39.95. It has previously been released in versions for the Commodore 64/128 and Apple II series.

Access Software Inc., 2561 South 1560 West, Woods Cross, UT 84087.

Circle Reader Service Number 205.

Munching On The Apple

Munchers and Troggles abound in the world of *Word Munchers,* an educational game for grades one through five from Minnesota Educational Computing Corporation. Players earn points by making their Munchers eat words with a particular vowel sound while avoiding the enemy Troggles. Teachers can determine which vowel sounds are used and can control the level of word difficulty. Approximately 1,700 words of varying difficulty are included.

Word Munchers runs on all Apple II computers with at least 64K RAM; joystick is optional. Suggested retail price, $49.

Minnesota Educational Computing Corporation, 3490 Lexington Avenue North, St. Paul, MI 55112.

Circle Reader Service Number 206.

Commodore Chemistry

Simon & Schuster has released a Commodore 64 version of the *Chem Lab* educational program for ages nine through twelve. The program contains 50 chemistry experiments with three levels of difficulty. All experiments are simulations of real experiments with actual results. Players can work their way up from Lab Assistant to Nobel Prize Winner.

The computerized laboratory comes equipped with on-screen simulations of: two robot arms for handling chemicals and equipment, five different pieces of lab equipment, plus three Bunsen burners and separate dispensers for gases, liquids, and solids. The chemical reactions are animated and change color, glow, melt, boil, and explode. On-screen messages tell the players what has been created.

Chem Lab for the Commodore 64, with its 96-page user's guide, sells for $39.95. Apple II and IBM PC/PCjr versions are also available.

Simon & Schuster Computer Software, 1230 Avenue of the Americas, New York, NY 10020.

Circle Reader Service Number 207.

Gandalf The Sorcerer For 64

A spellbound treasure is hidden in a castle surrounded by scaly tailed lizardmen. You, Gandalf the Sorcerer, must protect the treasure by using magic powers from a shining star. Such is the scenario of *Gandalf the Sorcerer,* Tymac's new adventure game for the Commodore 64. The game is for one player and requires a joystick. Three-dimensional graphics are featured.

Suggested retail price is $39.95.

Tymac Controls Corporation, 127 Main Street, Franklin, NJ 07416.

Circle Reader Service Number 208.

The lizardmen ambush the castle in Gandalf the Sorcerer.

Paper Airplane Kit

Simon & Schuster has released *The Great International Paper Airplane Construction Kit,* a set of paper airplane templates based on the bestselling book by the same name. The program contains over a dozen full-page paper airplane designs from biplanes to space shuttles. It also comes with a library of airplane graphics to embellish the airplanes with insignias, logos, windows, engines, pilots, and stewardesses. Also included is a step-by-step manual with instructions, suggestions, and a history of paper aviation.

The Great International Paper Airplane Construction Kit runs on the Apple II series with 64K RAM ($34.95); on the IBM PC, PC-XT, PC AT, and PCjr, with DOS 2.0 or higher and color/graphics card ($34.95); on the Macintosh with 128K RAM ($39.95); and on the Commodore 64 or 128 ($29.95).

Simon & Schuster, 1230 Avenue of the Americas, New York, NY 10020.

Circle Reader Service Number 209.

New From Mindscape

In *Dick Francis' High Stakes,* a new interactive text adventure from Mindscape, you are a wealthy English horse owner who must foil a sinister plot to cheat you. Based on the book by the popular mystery writer, Dick Francis, the game involves gambling and intrigue.

Also new from Mindscape are *The American Challenge: A Sailing Simulation,* which recreates the America's Cup sailing race, for one or two players; and *James Bond 007 Goldfinger,* an interactive text adventure involving travel, exotic weaponry, and the loves of the legendary 007.

Each game lists for $39.95 and runs on the Apple II and IBM PC computers.

Mindscape Inc., 3444 Dundee Road, Northbrook, IL 60062.

Circle Reader Service Number 210.

Educational Programs For Pre-School, High School

Grover and Ernie from "Sesame Street" enliven two new educational games from CBS Learning Systems. *Grover's Animal Adventures* takes preschoolers into four different animal environments: the African Grasslands, the Atlantic Ocean, a North American forest, and a barnyard. Children select animated animals and objects and place them in the appropriate environment on land, in water, or in the sky. In *Ernie's Big Splash,* children help Ernie find his Rubber Duckie by building a pathway that leads from the Duckie's soap dish into Ernie's bathtub. Both games are for ages four to six; each lists for $14.95.

CBS has also released *Mastering the ACT* (American College Testing Assessment), a self-paced preparation course for high school students that was developed by the National Association of Secondary School Principals. The program features full-length simulated ACT pre- and post-tests which provide self-scoring and detailed error analysis. Development exercises cover English, math, social studies, and natural sciences. For the Commodore 64/128 ($79.95), the Apple II series, and IBM PC and PCjr ($99.95 each).

CBS Learning Systems, One Fawcett Place, Greenwich, CT 06836.

Circle Reader Service Number 211.

MLX Machine Language Entry Program For Atari

Charles Brannon, Program Editor

MLX is a labor-saving utility that allows almost fail-safe entry of machine language programs published in COMPUTE!. *You need to know nothing about machine language to use MLX—it was designed for everyone.*

"MLX" is a new way to enter long machine language (ML) programs with a minimum of fuss. MLX lets you enter the numbers from a special list that looks similar to BASIC DATA statements. It checks your typing on a line-by-line basis. It won't let you enter illegal characters when you should be typing numbers. It won't let you enter numbers greater than 255 (forbidden in ML). It won't let you enter the wrong numbers on the wrong line. In addition, MLX creates a ready-to-use tape or disk file.

Using MLX

Type in and save MLX (you'll want to use it in the future). When you're ready to type in an ML program, run MLX. MLX asks you for three numbers: the starting address, the ending address, and the run/init address. These numbers are given in the article accompanying the ML program presented in MLX format. You must also choose one of three options for saving the file: as a boot tape, as disk binary file, or as boot disk. The article with the ML program should specify which formats may be used.

When you run MLX, you'll see a prompt corresponding to the starting address. The prompt is the current line you are entering from the listing. It increases by six each time you enter a line. That's because each line has seven numbers—six actual data numbers plus a checksum number. The checksum verifies that you typed the previous six numbers correctly. If you enter any of the six numbers wrong, or enter the checksum wrong, the computer rings a buzzer and prompts you to reenter the line. If you enter it correctly, a bell tone sounds and you continue to the next line.

MLX accepts only numbers as input. If you make a typing error, press the DEL/BACK SPACE; the entire number is deleted. You can press it as many times as necessary back to the start of the line. If you enter three-digit numbers as listed, the computer automatically prints the comma and goes on to accept the next number. If you enter fewer than three digits, you can press the comma key, the space bar, or the RETURN key to advance to the next number. The checksum automatically appears in inverse video for emphasis.

MLX Commands

When you finish typing an ML listing (assuming you type it all in one session), you can then save the completed program on tape or disk. Follow the screen instructions. If you get any errors while saving, you probably have a bad disk, or the disk is full, or you've made a typo when entering the MLX program itself.

You don't have to enter the whole ML program in one sitting. MLX lets you enter as much as you want, save it, and then reload the file from tape or disk later. MLX recognizes these commands:

CTRL-S	**Save**
CTRL-L	**Load**
CTRL-N	**New Address**
CTRL-D	**Display**

To issue a command, hold down the CTRL key (CONTROL on the XL models) and press the indicated key. When you enter a command, MLX jumps out of the line you've been typing, so we recommend you do it at a new prompt. Use the Save command (CTRL-S) to save what you've been working on. It will save on tape or disk, as if you've finished, but the tape or disk won't work, of course, until you finish the typing. Remember to make a note of what address you stop at. The next time you run MLX, answer all the prompts as you did before—regardless of where you stopped typing—then insert the disk or tape. When you get to the line number prompt, press CTRL-L to reload the partly completed file into memory. Then use the New Address command to resume typing.

To use the New Address command, press CTRL-N and enter the address where you previously stopped. The prompt will change, and you can then continue typing. Always enter a New Address that matches up with one of the line numbers in the MLX-format listing, or else the checksum won't work. The Display command lets you display a section of your typing. After you press CTRL-D, enter two addresses within the line number range of the listing. You can break out of the listing display and return to the prompt by pressing any key.

Atari MLX: Machine Language Entry

For instructions on entering this listing, please refer to "COMPUTE!'s Guide to Typing In Programs" in this issue of COMPUTE!.

```
DA 100 GRAPHICS 0:DL=PEEK(56
       0)+256*PEEK(561)+4:PO
       KE DL-1,71:POKE DL+2,
       6
NJ 110 POSITION 8,0:? "MLX":
       POSITION 23,0:? "[rest
       safe entry]":POKE 710,
       0:?
JK 120 ? "Starting Address";
       :INPUT BEG:? "  Endin
       g Address";:INPUT FIN
       :? "Run/Init Address"
       ;:INPUT STARTADR
DD 130 DIM A(6),BUFFER$(FIN-
       BEG+127),T$(20),F$(20
       ),CID$(7),SECTDR$(12B
       ),DSKINV$(6)
JJ 140 OPEN #1,4,0,"K:":? :?
       ,"[T]ape or [D]isk:";
BM 150 BUFFER$=CHR$(0):BUFFE
       R$(FIN-BEG+30)=BUFFER
       $:BUFFER$(2)=BUFFER$:
       SECTDR$=BUFFER$
GC 160 ADDR=BEG:CID$="hhh":C
       ID$(4)=CHR$(170):CID$
       (5)="LV":CID$(7)=CHR$
       (22B)
EJ 170 GET #1,MEDIA:IF MEDIA
       <>B4 AND MEDIA<>6B TH
       EN 170
PO 180 ? CHR$(MEDIA):? :IF M
       EDIA<>ASC("T") THEN B
       UFFER$="":GOTO 250
PL 190 BEG=BEG-24:BUFFER$=CH
       R$(0):BUFFER$(2)=CHR$
       (INT((FIN-BEG+127)/12
       B))
KF 200 H=INT(BEG/256):L=BEG-
       H*256:BUFFER$(3)=CHR$
       (L):BUFFER$(4)=CHR$(H
       )
EC 210 PINIT=BEG+B:H=INT(PIN
       IT/256):L=PINIT-H*256
       :BUFFER$(5)=CHR$(L):B
       UFFER$(6)=CHR$(H)
PB 220 FOR I=7 TO 24:READ A:
       BUFFER$(I)=CHR$(A):NE
       XT I:DATA 24,96,169,6
       0,141,2,211,169,0,133
       ,10,169,0,133,11,76,0
       ,0
DP 230 H=INT(STARTADR/256):L
       =STARTADR-H*256:BUFFE
       R$(15)=CHR$(L):BUFFER
       $(19)=CHR$(H)
KL 240 BUFFER$(23)=CHR$(L):B
       UFFER$(24)=CHR$(H)
HI 250 IF MEDIA<>ASC("D") TH
       EN 360
OO 260 ? :? "Boot [D]isk or Bi
       nary [F]ile:";
LI 270 GET #1,DTYPE:IF DTYPE
```

```
         <>68 AND OTYPE<>70 TH
         EN 270
BM 280 ? CHR$(DTYPE):IF OTYP
         E=70 THEN 360
PJ 290 BEG=BEG-30:BUFFER$=CH
         R$(0):BUFFER$(2)=CHR$
         (INT((FIN-BEG+127)/12
         B))
KB 300 H=INT(BEG/256):L=BEG-
         H*256:BUFFER$(3)=CHR$
         (L):BUFFER$(4)=CHR$(H
         )
HH 310 PINIT=STARTAOR:H=INT(
         PINIT/256):L=PINIT-H*
         256:BUFFER$(5)=CHR$(L
         ):BUFFER$(6)=CHR$(H)
AD 320 RESTORE 330:FOR I=7 T
         O 30:READ A:BUFFER$(I
         )=CHR$(A):NEXT I
BA 330 DATA 169,0,141,231,2,
         133,14,169,0,141,232,
         2,133,15,169,0,133,10
         ,169,0,133,11,24,96
OB 340 H=INT(BEG/256):L=BEG-
         H*256:BUFFER$(B)=CHR$
         (L):BUFFER$(15)=CHR$(
         H)
OD 350 H=INT(STARTAOR/256):L
         =STARTADR-H*256:BUFFE
         R$(22)=CHR$(L):BUFFER
         $(26)=CHR$(H)
JP 360 GRAPHICS 0:POKE 712,1
         0:PDKE 710,10:POKE 70
         9,2
JK 370 ? ADOR;":";:FOR J=1 T
         O 6
NF 380 GOSUB 570:IF N=-1 THE
         N J=J-1:GOTO 380
BF 390 IF N=-19 THEN 720
OI 400 IF N=-12 THEN LET REA
         O=1:GDTO 720
AI 410 TRAP 410:IF N=-14 THE
         N ? :? "New Address";
         :INPUT AOOR:? :GDTO 3
         70
JD 420 TRAP 32767:IF N<>-4 T
         HEN 480
AJ 430 TRAP 430:? :? "OisplI
         y:From";:INPUT F:? ,"
         To";:INPUT T:TRAP 327
         67
ML 440 IF F<BEG DR F>FIN DR
         T<BEG DR T>FIN DR T<F
          THEN ? CHR$(253);"At
          Ieast ";BEG;", Not M
         ore Than ";FIN:GDTD 4
         30
MH 450 FDR I=F TD T STEP 6:?
          :? I;":";:FDR K=0 TD
          5:N=PEEK(ADR(BUFFER$
         )+1+K-BEG):T$="000":T
         $(4-LEN(STR$(N)))=STR
         $(N)
MA 460 IF PEEK(764)<255 THEN
          GET #1,A:POP :POP :?
          :GDTD 370
FM 470 ? T$;",";:NEXT K:? CH
         R$(126);:NEXT 1:? :?
         :GDTD 370
BA 480 IF N<0 THEN ? :GOTD 3
         70
MH 490 A(J)=N:NEXT J
JM 500 CKSUM=AODR-INT(ADOR/2
         56)*256:FDR I=1 TD 6:
         CKSUM=CKSUM+A(I):CKSU
         M=CKSUM-256*(CKSUM>25
         5):NEXT I
KK 510 RF=128:SDUND 0,200,12
         ,8:GDSUB 570:SDUNO 0,
         0,0,0:RF=0:? CHR$(126
         )
CM 520 IF N<>CKSUM THEN ? :?
          "Incorrect";CHR$(253
         );:? :GOTO 370
EK 530 FDR W=15 TD 0 STEP -1
         :SOUNO 0,50,10,W:NEXT
          W
FL 540 FOR I=1 TD 6:POKE AOR
         (BUFFER$)+AOOR-BEG+I-
         1,A(I):NEXT I
HS 550 AOOR=AOOR+6:IF AOOR<=
         FIN THEN 370
GM 560 GOTO 710
FI 570 N=0:Z=0
PN 580 GET #1,A:IF A=155 DR
         A=44 OR A=32 THEN 670
FB 590 IF A<32 THEN N=-A:RET
         URN
EB 600 IF A<>126 THEN 630
ML 610 GOSUB 690:IF I=1 AND
         T=44 THEN N=-1:? CHR$
         (126);:GOTO 690
GN 620 GOTD 570
GJ 630 IF A<4B DR A>57 THEN
         5B0
AN 640 ? CHR$(A+RF);:N=N*10+
         A-4B
EB 650 IF N>255 THEN ? CHR$(
         253);:A=126:GOTO 600
EH 660 Z=Z+1:IF Z<3 THEN 5B0
JH 670 IF Z=0 THEN ? CHR$(25
         3);:GOTO 570
KC 680 ? ",";:RETURN
NO 690 POKE 752,1:FOR I=1 TD
          3:? CHR$(30);:GET #6
         ,T:IF T<>44 AND T<>5B
          THEN ? CHR$(A);:NEXT
          I
PI 700 POKE 752,0:? " ";CHR$
         (126);:RETURN
KM 710 GRAPHICS 0:PDKE 710,2
         6:POKE 712,26:POKE 70
         9,2
FF 720 IF MEOIA=ASC("T") THE
         N B90
OJ 730 REM DISK
OK 740 IF READ THEN ? :? "Lo
         ad File":?
IB 750 IF DTYPE<>70 THEN 104
         0
AE 760 ? :? "Enter AUTDRUN.S
         YS for automatic use"
         :? :? "Enter filename
         ":INPUT T$
GF 770 F$=T$:IF LEN(T$)>2 TH
         EN IF T$(1,2)<>"O:" T
         HEN F$="D:":F$(3)=T$
NJ 780 TRAP B70:CLDSE #2:DPE
         N #2,8-4*READ,0,F$:?
         :? "Working..."
JM 790 IF READ THEN FDR I=1
         TD 6:GET #2,A:NEXT I:
         GDTQ 820
PO B00 PUT #2,255:PUT #2,255
OJ B10 H=INT(BEG/256):L=BEG-
         H*256:PUT #2,L:PUT #2
         ,H:H=INT(FIN/256):L=F
         IN-H*256:PUT #2,L:PUT
          #2,H
NF B20 GDSUB 970:IF PEEK(195
         )>1 THEN B70
IF B30 IF STARTADR=0 OR READ
          THEN B50
FO B40 PUT #2,224:PUT #2,2:P
         UT #2,225:PUT #2,2:H=
         INT(STARTADR/256):L=S
         TARTAOR-H*256:PUT #2,
         L:PUT #2,H
MM B50 TRAP 32767:CLDSE #2:?
          "Finished.":IF READ
         THEN ? :? :LET READ=0
         :GDTD 360
HF B60 END
FO B70 ? "Error ";PEEK(195);
         " trying to access":?
          F$:CLOSE #2:? :GOTD
         760
MC 880 REM BOOT TAPE
HN B90 IF READ THEN ? :? "Re
         ad Tape"
HI 900 ? :? :? "Insert, Rewi
         nd Tape.":? "Press PL
         AY ";:IF  NOT READ TH
         EN ? "& RECORO"
LP 910 ? :? "Press RETURN wh
         en ready:";
JH 920 TRAP 960:CLOSE #2:DPE
         N #2,8-4*READ,128,"C:
         ":? :? "Working..."
NH 930 GDSUB 970:IF PEEK(195
         )>1 THEN 960
HM 940 CLOSE #2:TRAP 32767:?
          "Finished.":? :? :IF
          READ THEN LET READ=0
         :GOTO 360
HF 950 ENO
CD 960 ? :? "Error ";PEEK(19
         5);" when reading/wri
         ting boot tape":? :CL
         OSE #2:GOTO 890
MB 970 REM CIO Load/Save Fil
         e#2 opened READ=0 fo
         r write, READ=1 for r
         ead
EA 980 X=32:REM File#2,$20
EF 990 ICCOM=834:ICBAOR=836:
         ICBLEN=840:ICSTAT=835
MD 1000 H=INT(AOR(BUFFER$)/2
         56):L=AOR(BUFFER$)-H
         *256:POKE ICBAOR+X,L
         :POKE ICBAOR+X+1,H
FH 1010 L=FIN-BEG+1:H=INT(L/
         256):L=L-H*256:POKE
         ICBLEN+X,L:POKE ICBL
         EN+X+1,H
MD 1020 POKE ICCOM+X,11-4*RE
         AD:A=USR(ADR(CID$),X
         )
BG 1030 PDKE 195,PEEK(ICSTAT
         ):RETURN
KA 1040 REM SECTOR I/O
GC 1050 IF READ THEN 1100
HE 1060 ? :? "Format Disk In
          Drive 1? (Y/N):";
FC 1070 GET #1,A:IF A<>78 AN
         D A<>89 THEN 1070
EC 1080 ? CHR$(A):IF A=78 TH
         EN 1100
CP 1090 ? :? "Formatting..."
         :XIO 254,#2,0,0,"O:"
         :? "Format Complete"
         :?
AC 1100 NR=INT((FIN-8EG+127)
         /128):BUFFER$(FIN-BE
         G+2)=CHR$(0):IF READ
          THEN ? "Reading..."
         :GOTD 1120
LE 1110 ? "Writing..."
LI 1120 FDR I=1 TD NR:S=I
IO 1130 IF READ THEN GDSUB 1
         220:BUFFER$(I*128-12
         7)=SECTDR$:GDTD 1160
PL 1140 SECTDR$=BUFFER$(I*12
         B-127)
AM 1150 GDSUB 1220
DM 1160 IF PEEK(DSTATS)<>1 T
         HEN 1200
FB 1170 NEXT I
GM 1180 IF  NDT READ THEN EN
         D
DH 1190 ? :? :LET READ=0:GDT
         D 360
JJ 1200 ? "Error on disk acc
         ess.":? "May need fo
         rmatting.":GDTD 1040
KI 1210 REM
```

```
BL 1220  REM  SECTOR ACCESS S
         UBROUTINE
IB 1230  REM Drive ONE
IH 1240  REM Pass buffer in S
         ECTOR$
MP 1250  REM sector # in vari
         able S
EG 1260  REM READ=1 for read,
KJ 1270  REM READ=0 for write
BN 1280  BASE=3*256
GL 1290  DUNIT=BASE+1:DCOMND=
         BASE+2:DSTATS=BASE+3
NL 1300  DBUFLO=BASE+4:DBUFHI
         =BASE+5
AI 1310  DBYTLO=BASE+8:DBYTHI
         =BASE+9
JA 1320  DAUX1=BASE+10:DAUX2=
         BASE+11
PN 1330  REM DIM DSKINV$(4)
CA 1340  DSKINV$="hLS":DSKINV
         $(4)=CHR$(228)
PF 1350  POKE DUNIT,1:A=ADR(S
         ECTOR$):H=INT(A/256)
         :L=A-256*H
BK 1360  POKE DBUFHI,H
CO 1370  POKE DBUFLO,L
PD 1380  POKE DCOMND,87-5*REA
         D
AA 1390  POKE DAUX2,INT(S/256
         ):POKE DAUX1,S-PEEK(
         DAUX2)*256
KJ 1400  A=USR(ADR(DSKINV$))
KB 1410  RETURN
```

©

COMPUTE!'s Guide To Typing In Programs

Computers are precise—type the program *exactly* as listed, including necessary punctuation and symbols, except for special characters noted below. We have implemented a special listing convention as well as a program to check your typing—"Automatic Proofreader.

Commodore, Apple, and Atari programs can contain some hard-to-read special characters, so we have a listing system that indicates these control characters. You will find these Commodore and Atari characters in curly braces; *do not type the braces.* For example, {CLEAR} or {CLR} instructs you to insert the symbol which clears the screen on the Atari or Commodore machines. For Commodore, Apple, and Atari, a symbol by itself within curly braces is usually a control key or graphics key. If you see {A}, hold down the CONTROL key and press A. This will produce a reverse video character on the Commodore (in quote mode), a graphics character on the Atari, and an invisible control character on the Apple. Graphics characters entered with the Commodore logo key are enclosed in a special bracket: [<A>]. In this case, you would hold down the Commodore logo key as you type A. Our Commodore listings are in uppercase, so shifted symbols are underlined. A graphics heart symbol (SHIFT-S) would be listed as S. One exception is {SHIFT-SPACE}. When you see this, hold down SHIFT and press the space bar. If a number precedes a symbol, such as {5 RIGHT}, {6 S}, or [<8 Q>], you would enter five cursor rights, six shifted S's, or eight Commodore-Q's. On the Atari, inverse characters (white on black) should be entered with the Atari logo key.

Any more than two spaces will be listed. For example, {6 SPACES} means press the space bar six times. Our listings never leave a space at the end of a line, instead moving it to the next printed line as {SPACE}.

Atari 400/800/XL/XE

When you see	Type	See	
{CLEAR}	ESC SHIFT <		Clear Screen
{UP}	ESC CTRL -	↑	Cursor Up
{DOWN}	ESC CTRL =	↓	Cursor Down
{LEFT}	ESC CTRL +	←	Cursor Left
{RIGHT}	ESC CTRL *	→	Cursor Right
{BACK S}	ESC DELETE		Backspace
{DELETE}	ESC CTRL DELETE		Delete character
{INSERT}	ESC CTRL INSERT		Insert character
{DEL LINE}	ESC SHIFT DELETE		Delete line
{INS LINE}	ESC SHIFT INSERT		Insert line
{TAB}	ESC TAB		TAB key
{CLR TAB}	ESC CTRL TAB		Clear tab
{SET TAB}	ESC SHIFT TAB		Set tab stop
{BELL}	ESC CTRL 2		Ring buzzer
{ESC}	ESC ESC		ESCape key

Commodore PET/CBM/VIC/64/128/16/+4

When You Read:	Press:	See:
{CLR}	SHIFT + CLR/HOME	
{HOME}	CLR/HOME	
{UP}	SHIFT + ↑ CRSR ↓	
{DOWN}	↑ CRSR ↓	
{LEFT}	SHIFT + ← CRSR →	
{RIGHT}	← CRSR →	
{RVS}	CTRL + 9	
{OFF}	CTRL + 0	
{BLK}	CTRL + 1	
{WHT}	CTRL + 2	
{RED}	CTRL + 3	
{CYN}	CTRL + 4	
{PUR}	CTRL + 5	
{GRN}	CTRL + 6	
{BLU}	CTRL + 7	
{YEL}	CTRL + 8	

When You Read:	Press:	See:
[<1>]	COMMODORE + 1	
[<2>]	COMMODORE + 2	
[<3>]	COMMODORE + 3	
[<4>]	COMMODORE + 4	
[<5>]	COMMODORE + 5	
[<6>]	COMMODORE + 6	
[<7>]	COMMODORE + 7	
[<8>]	COMMODORE + 8	
{F1}	f1	
{F2}	SHIFT + f1	
{F3}	f3	
{F4}	SHIFT + f3	
{F5}	f5	
{F6}	SHIFT + f5	
{F7}	f7	
{F8}	SHIFT + f7	
←	←	

The Automatic Proofreader

This month, we are featuring a completely new Proofreader for the Commodore 64, 128, VIC, Plus/4, and 16. Please refer to "The New Automatic Proofreader for Commodore" article elsewhere in this issue for more information. Type in the appropriate program listed below, then save it for future use. On the Atari, run the Proofreader to activate it, then enter NEW to erase the BASIC loader (the Proofreader remains active in memory as a machine language program). Pressing SYSTEM RESET deactivates the Proofreader. Use PRINT USR(1536) to reenable the Atari Proofreader. The Apple Proofreader erases the BASIC portion of itself after you RUN it, leaving only the machine language portion in memory. It works with either DOS 3.3 or ProDOS. Disable the Apple Proofreader by pressing CTRL-RESET before running another BASIC program. The IBM Proofreader is a BASIC program that simulates the IBM BASIC line editor, letting you enter, edit, list, save, and load programs that you type. Type RUN to activate.

Once the Proofreader is active, try typing in a line. As soon as you press RETURN, a hexadecimal number (on the Apple) or a pair of letters (on the Atari or IBM) appears. The number or pair of letters is called a *checksum*.

Compare the value provided by the Proofreader with the checksum printed in the program listing in the magazine. In Commodore listings, the checksum is set off from the rest of the line with *rem*. This prevents a syntax error if the checksum is typed in, but the REM statements and checksums need *not* be typed in.

In Atari, Apple, and IBM listings, the checksum is given to the left of each line number. Just type in the program, a line at a time (without the printed checksum) and compare the checksums. If they match, go on to the next line. If not, check your typing: You've made a mistake. On the Atari and Apple Proofreaders, spaces are not counted as part of the checksum, so be sure you type the right number of spaces between quote marks. The Atari Proofreader does not check to see that you've typed the characters in the right order, so if characters are transposed, the checksum stil matches the listing. Because of the checksum method used, do not use abbreviations, such as ? for PRINT. The IBM Proofreader is the pickiest of all; it *will* detect errors in spacing and transposition. Be sure to leave Caps Lock on, except when typing lowercase characters.

IBM Proofreader Commands

Since the IBM Proofreader (Program 2) replaces the computer's normal BASIC line editor, it has to include many of the direct-mode IBM BASIC commands. The syntax is identical to IBM BASIC. Commands simulated are LIST, LLIST, NEW, FILES, SAVE, and LOAD. When listing your program, press any key (except Ctrl-Break) to stop the listing. If you enter NEW, the Proofreader will prompt you to press Y to be especially sure you mean yes.

Two new commands are BASIC and CHECK. BASIC exits the Proofreader back to IBM BASIC, leaving the Proofreader in memory. CHECK works just like LIST, but shows the checksums along with the listing. After you have typed in a program, save it to disk. Then exit the Proofreader with the BASIC command, and load the program into the normal BASIC environment (this will replace the Proofreader in memory). You can now run the program, but you may want to resave it to disk. This will shorten it on disk and make it load faster, but it can no longer be edited with the Proofreader. If you want to convert a program to Proofreader format, save it to disk with SAVE "filename",A.

Program 1: Atari Proofreader

By Charles Brannon, Program Editor

```
100 GRAPHICS 0
110 FOR I=1536 TO 1700:RE
    AD A:POKE I,A:CK=CK+A
    :NEXT I
120 IF CK<>19072 THEN ? "
    Error in DATA Stateme
    nts.  Check Typing.":
    END
130 A=USR(1536)
140 ? :? "Automatic Proof
    reader Now Activated.
    "
150 END
160 DATA 104,160,0,185,26
    ,3,201,69,240,7
170 DATA 200,200,192,34,2
    08,243,96,200,169,74
180 DATA 153,26,3,200,169
    ,6,153,26,3,162
190 DATA 0,189,0,228,157,
    74,6,232,224,16
200 DATA 208,245,169,93,1
    41,78,6,169,6,141
210 DATA 79,6,24,173,4,22
    8,105,1,141,95
220 DATA 6,173,5,228,105,
    0,141,96,6,169
230 DATA 0,133,203,96,247
    ,238,125,241,93,6
240 DATA 244,241,115,241,
    124,241,76,205,238
250 DATA 0,0,0,0,0,32,62,
    246,8,201
260 DATA 155,240,13,201,3
    2,240,7,72,24,101
270 DATA 203,133,203,104,
    40,96,72,152,72,138
280 DATA 72,160,0,169,128
    ,145,88,200,192,40
290 DATA 208,249,165,203,
    74,74,74,74,24,105
300 DATA 161,160,3,145,88
    ,165,203,41,15,24
310 DATA 105,161,200,145,
    88,169,0,133,203,104
320 DATA 170,104,168,104,
    40,96
```

Program 2: IBM Proofreader

By Charles Brannon, Program Editor

```
MC 10 'Automatic Proofreader Ver
      sion 3.0 (Lines 205,206 ad
      ded/190 deleted/470,490 ch
      anged from V2.0)
LD 100 DIM L$(500),LNUM(500):COL
       OR 0,7,7:KEY OFF:CLS:MAX=
       0:LNUM(0)=65536!
PK 110 ON ERROR GOTO 120:KEY 15,
       CHR$(4)+CHR$(70):ON KEY(1
       5) GOSUB 640:KEY (15) ON:
       GOTO 130
BE 120 RESUME 130
BJ 130 DEF SEG=&H40:W=PEEK(&H4A)
IN 140 ON ERROR GOTO 650:PRINT:P
       RINT"Proofreader Ready."
KB 150 LINE INPUT L$:Y=CSRLIN-IN
       T(LEN(L$)/W)-1:LOCATE Y,1
CA 160 DEF SEG=0:POKE 1050,30:PO
       KE 1052,34:POKE 1054,0:PO
       KE 1055,79:POKE 1056,13:P
       OKE 1057,28:LINE INPUT L$
       :DEF SEG:IF L$="" THEN 15
       0
BC 170 IF LEFT$(L$,1)=" " THEN L
       $=MID$(L$,2):GOTO 170
NH 180 IF VAL(LEFT$(L$,2))=0 AND
        MID$(L$,3,1)=" " THEN L$
       =MID$(L$,4)
ND 200 IF ASC(L$)>57 THEN 260 'n
       o line number, therefore
       command
JB 205 BL=INSTR(L$," "):IF BL=0
       THEN BL$=L$:GOTO 206 ELSE
        BL$=LEFT$(L$,BL-1)
GN 206 LNUM=VAL(BL$):TEXT$=MID$(
       L$,LEN(STR$(LNUM))+1)
OG 210 IF TEXT$="" THEN GOSUB 54
       0:IF LNUM=LNUM(P) THEN GO
       SUB 560:GOTO 150 ELSE 150
NB 220 CKSUM=0:FOR I=1 TO LEN(L$
       ):CKSUM=(CKSUM+ASC(MID$(L
       $,I))*I) AND 255:NEXT:LOC
       ATE Y,1:PRINT CHR$(65+CKS
       UM/16)+CHR$(65+(CKSUM AND
        15))+" "+L$
JE 230 GOSUB 540:IF LNUM(P)=LNUM
        THEN L$(P)=TEXT$:GOTO 15
       0 'replace line
CL 240 GOSUB 580:GOTO 150 'inser
       t the line
AD 260 TEXT$="":FOR I=1 TO LEN(L
       $):A=ASC(MID$(L$,I)):TEXT
       $=TEXT$+CHR$(A+32*(A>96 A
       ND A<123)):NEXT
LP 270 DELIMITER=INSTR(TEXT$," "
       ):COMMAND$=TEXT$:ARG$="":
       IF DELIMITER THEN COMMAND
       $=LEFT$(TEXT$,DELIMITER-1
       ):ARG$=MID$(TEXT$,DELIMIT
       ER+1) ELSE DELIMITER=INST
       R(TEXT$,CHR$(34)):IF DELI
       MITER THEN COMMAND$=LEFT$
       (TEXT$,DELIMITER-1):ARG$=
       MID$(TEXT$,DELIMITER)
FC 280 IF COMMAND$<>"LIST" THEN
       410
ID 290 OPEN "scrn:" FOR OUTPUT A
       S #1
LN 300 IF ARG$="" THEN FIRST=0:P
       =MAX-1:GOTO 340
```

```
IJ 310 DELIMITER=INSTR(ARG$,"-")
       :IF DELIMITER=0 THEN LNUM
       =VAL(ARG$):GOSUB 540:FIRS
       T=P:GOTO 340
BP 320 FIRST=VAL(LEFT$(ARG$,DELI
       MITER)):LAST=VAL(MID$(ARG
       $,DELIMITER+1))
EC 330 LNUM=FIRST:GOSUB 540:FIRS
       T=P:LNUM=LAST:GOSUB 540:I
       F P=0 THEN P=MAX-1
GD 340 FOR X=FIRST TO P:N$=MID$(
       STR$(LNUM(X)),2)+" "
KA 350 IF CKFLAG=0 THEN A$="":GO
       TO 370
PF 360 CKSUM=0:A$=N$+L$(X):FOR I
       =1 TO LEN(A$):CKSUM=(CKSU
       M+ASC(MID$(A$,I))*I) AND
       255:NEXT:A$=CHR$(65+CKSUM
       /16)+CHR$(65+(CKSUM AND 1
       5))+" "
DO 370 PRINT #1,A$+N$+L$(X)
JJ 380 IF INKEY$<>"" THEN X=P
DF 390 NEXT :CLOSE #1:CKFLAG=0
CA 400 GOTO 130
PD 410 IF COMMAND$="LLIST" THEN
       OPEN "lpt1:" FOR OUTPUT A
       S #1:GOTO 300
BM 420 IF COMMAND$="CHECK" THEN
       CKFLAG=1:GOTO 290
KA 430 IF COMMAND$<>"SAVE" THEN
       450
CL 440 GOSUB 600:OPEN ARG$ FOR O
       UTPUT AS #1:ARG$="":GOTO
       300
DE 450 IF COMMAND$<>"LOAD" THEN
       490
PG 460 GOSUB 600:OPEN ARG$ FOR I
       NPUT AS #1:MAX=0:P=0
KA 470 WHILE NOT EOF(1):LINE INP
       UT #1,L$:BL=INSTR(L$," ")
       :BL$=LEFT$(L$,BL-1):LNUM(
       P)=VAL(BL$):L$(P)=MID$(L$
       ,LEN(BTR$(VAL(BL$)))+1):P
       =P+1:WEND
KK 480 MAX=P:CLOSE #1:GOTO 130
GJ 490 IF COMMAND$="NEW" THEN IN
       PUT "Erase program - Are
       you sure";L$:IF LEFT$(L$,
       1)="y" OR LEFT$(L$,1)="Y"
        THEN MAX=0:LNUM(0)=65536
       !:GOTO 130:ELSE 130
CL 500 IF COMMAND$="BASIC" THEN
       CDLDR 7,0,0:ON ERROR GOTO
       0:CLS:END
NC 510 IF COMMAND$<>"FILES" THEN
       520
IH 515 IF ARG$="" THEN ARG$="A:"
       ELSE SEL=1:GOSUB 600
ID 517 FILES ARG$:GOTO 130
DO 520 PRINT"Syntax error":GOTO
       130
GD 540 P=0:WHILE LNUM>LNUM(P) AN
       D P<MAX:P=P+1:WEND:RETURN
KH 560 MAX=MAX-1:FOR X=P TO MAX:
       LNUM(X)=LNUM(X+1):L$(X)=L
       $(X+1):NEXT:RETURN
GK 580 MAX=MAX+1:FOR X=MAX TO P+
       1 STEP -1:LNUM(X)=LNUM(X-
       1):L$(X)=L$(X-1):NEXT:L$(
       P)=TEXT$:LNUM(P)=LNUM:RET
       URN
BA 600 IF LEFT$(ARG$,1)<>CHR$(34
       ) THEN 520 ELSE ARG$=MID$
       (ARG$,2)
EE 610 IF RIGHT$(ARG$,1)=CHR$(34
       ) THEN ARG$=LEFT$(ARG$,LE
       N(ARG$)-1)
LA 620 IF SEL=0 AND INSTR(ARG$,"
       .")=0 THEN ARG$=ARG$+".BA
       B"
DD 630 SEL=0:RETURN
HK 640 CLOSE #1:CKFLAG=0:PRINT"B
       topped.":RETURN 150
II 650 PRINT "Error #";ERR:REBUM
       E 150
```

Program 3: Apple Proofreader

By Tim Victor, Editorial Programmer

```
10 C = 0: FOR I = 768 TO 768 +
   68: READ A:C = C + A: POKE I
   ,A: NEXT
20 IF C < > 7258 THEN PRINT "ER
   ROR IN PROOFREADER DATA STAT
   EMENTS": END
30 IF PEEK (190 * 256) < > 76 T
   HEN POKE 56,0: POKE 57,3: CA
   LL 1002: GOTO 50
40 PRINT CHR$ (4);"IN#A$300"
50 POKE 34,0: HOME : POKE 34,1:
    VTAB 2: PRINT "PROOFREADER
   INSTALLED"
60 NEW
100 DATA 216,32,27,253,201,141
110 DATA 208,60,138,72,169,0
120 DATA 72,189,255,1,201,160
130 DATA 240,8,104,10,125,255
140 DATA 1,105,0,72,202,208
150 DATA 238,104,170,41,15,9
160 DATA 48,201,58,144,2,233
170 DATA 57,141,1,4,138,74
180 DATA 74,74,74,41,15,9
190 DATA 48,201,58,144,2,233
200 DATA 57,141,0,4,104,170
210 DATA 169,141,96
```

©

COMPUTE!

SpeedScript Update

There is an error in the correction to Apple *SpeedScript* from the *"SpeedScript 3.0* Revisited" article in the December 1985 issue (p. 90) which causes the page number to repeat continuously when the # formatting command is used. In line 1C88 of the listing, the 9D should be a 9C. Load *SpeedScript* back into Apple "MLX" and enter the following replacement line:

1C88: AC E5 1E D0 9C AE E6 1E EC

After making the correction, be sure to use the MLX Save option to save a new copy of *SpeedScript*.

The item in the January 1986 "Reader's Feedback" column (p. 10) that told how to make Commodore 64 *SpeedScript 3.0* default to disk for saving and loading had transposed digits in the middle POKE address. The line should have read:

POKE 4904,234: POKE 4905,169: POKE 4906,68

This modification works for all updates of version 3 (3.0, 3.1, or 3.2).

Atari Solitaire

The Atari listing for this game from the January 1986 issue (Program 2, p. 48) has a typographical error in line 910. The third character in S$, which defines the card suits, should be {.} instead of the apostrophe shown. CTRL-period is the diamond graphic character.

Formatted Printouts For Commodore

There are two errors in the DATA statements for this program from the January 1986 issue (p. 99). In line 540, the null string, "", should come before the item "BLACK" rather than after it. In line 640, the last item in the line should be "↑" rather than a null string.

Skyscape For IBM & Apple

Certain combinations of date and time inputs cause syntax errors in the IBM and Apple versions of this astronomy program from the November 1985 issue (p. 62). To correct this, change CC <= 0 to CC <= 1 in line 2060 of the IBM version (Program 3) and line 1770 of the Apple version (Program 4).

Memo Diary

The Commodore version of this calendar utility from the December 1985 issue (p. 65) won't work with tape. Tape users should modify the OPEN statement in line 3170 as follows:

OPEN 1,1+7*D1,8*D1+1,F$+G$:

Author Jim Butterfield also recommends that line 660 be replaced with 660 REM. With this change the calendar file will always be updated. ©

Classified

SOFTWARE

TI-99/4A NEW STATES AND CAPITALS GAME
Hi-Res map of USA. Send $12 for cass.
Or $1 for more info. to: TRINITY SYSTEMS
1022 Grandview, Pittsburgh, PA 15237

TI-99/4A Software/Hardware bargains.
Hard-to-find items. Huge selection.
Fast service. Free catalog. D.E.C.,
Box 690, Hicksville, NY 11801

PROGRAMS FOR THE TANDY 1000
Send $1 for list of educat. & entertainment programs. Refundable with first purchase.
SODA POP SW, POB 653, Kenosha, WI 53141

FREE APPLE SOFTWARE
Over 1,000 Public Domain Programs on 50 diskettes, $5 each, plus $1 shipping per order. Send $1 for catalog, refundable.
C & H ENTERPRISES
Box 29243, Memphis, TN 38127

LOTTO PICKER. Improve your chances for those Million Dollar Jackpots! Picks LOTTO, WIN-4, & Daily Numbers. All USA & Can. games incl. Expandable! IBM/C64/TI99 $29.95. Order Now! 1-800-341-1950 Ext. 77. Mail Orders: Ridge, 170 Broadway, #201-C, NYC, NY 10038. Catalog.

SAVE MONEY! EASY TAX SIMPLIFIES THE 15 most common IRS tax forms. Faster, line by line preparation on screen and printer. Commodore 64, disk. Send $39.95 plus $2 s/h to Hybrid Software, 1739 Schilder Lane, Waverly, OH 45690

PROJECT PLANNING/MANAGEMENT using the C64, SX, or C128. Data sheet for SASE - Prgm for $106.95 (CA res. add 6% sls tx) LAWCO, Dept. C, Box 2009, Manteca, CA 95336

Genealogy Program for the C64. **"FAMILY TREE"** will produce Pedigree Charts, Family Group Records, Individual Files, Indexes, Searches of Ancestors. LDS version available. "The Best" genealogy program for the 64. $49.95, GENEALOGY SOFTWARE, POB 1151, PORT HURON, MI 48061, (519) 344-3990.

Animal Records maintained with **"PETIGREE"** for the C64. Produces Litter, Awards, Breeding Show, Individual Records, Pedigree Charts. $69.95. GENEALOGY SOFTWARE, POB 1151, PORT HURON, MI 48061, (519) 344-3990.

FREE SOFTWARE CATALOG!
Call Toll-Free 1-800-554-1162, Tevex, Inc.
Save ⅓ off retail prices. We carry SSI, Elect. Arts, Infocom, and many more!

COMMODORE. TRY BEFORE YOU BUY. Top 25 best-selling games, utilities, new releases. Visa, MasterCard. Free brochure. Rent-A-Disk, 908 9th Ave., Huntington, WV 25701 (304) 522-1665

INTEREST CALCULATIONS.
MAI-2 10 lets your computer help analyze investment decisions. Calc. future value, present value, annuities, sinking funds, loan pymt. sched., + more! Menu driven/Simple. IBM ck/mo. Munier Associates, Inc., Dept. A5, P.O. Box 79314, Houston, Texas 77279 (713) 784-4348.

*** * IBM-PCjr. OWNERS * ***
Learn to unleash jr's hidden powers!
How-to info from jr. experts, software tips, freeware, best buys and more! Send $1.50 for single issue or $18/yr. (12 issues) to:
JR. NEWSLETTER, Crider Associates,
Box 163, Southbury, CT 06488

FREE PROGRAMS! Apple/IBM PC/ TI99/ Timex/C64/+4/16/V20/Adam/TRS80 III/ 4/100/200/CoCo/PC3/MC10/PCjr. Send stamps! EZRAEZRA, Box 5222 TM, San Diego, CA 92105

TEACHERS - MAGNUM GRADEBOOK prepares student reports of assignments, scores, averages. J. KNOWLES, 1025 Darling St., Ogden, Utah 84403. Specify Commodore 64 or Apple IIe. Check or Visa: $29.

DISCOUNT SOFTWARE: Amiga/Apple/ Atari/C64-128/IBM PC-PCjr/TRS-80/Timex/ Sinclair. Free Catalog. WMJ DAT SYSTEMS, 4 Butterfly Dr., Hauppauge, NY 11788

MISCELLANEOUS

HELP IS ON THE WAY!
Just call 1-800-334-0868 to get your free copy of the latest COMPUTE! Books Catalog! If you need help in getting information on all of the latest COMPUTE! book titles available plus all COMPUTE! backlist titles, call us today!

Maxell MD1, $1.29-MD2, $1.99. Dysan 104/1D, $1.79-104/2D, $2.39. Shipping $3.75. Also Verbatim, IBM, 3M, BASF. TAPE WORLD, 220 Spring St., Butler, PA 16001, 1-800-245-6000. Visa, MC.

DISK SALE! - SS/DD 35-trk for Apple w/sleeve & label-10/$5.80, bulk-100/$45. Standard SS/DD w/sleeve & label-10/$7.50, bulk-100/$59. DS/DD w/sleeve & label-10/$8.50, bulk-100/$67. 3½" SS for Mac-10/$19.99. PREMIUM QUALITY, LIFETIME WARRANTY! Money-back satisfaction guarantee! Min. order $20. Send check or pay by MC/VISA/AE $3 shipping, + $2 if C.O.D. - UNITECH, 20 Hurley St., Cambridge, MA 02141. (800) 343-0472, in Mass. (617) "UNI-TECH".

EARN MONEY, PART OR FULL TIME, AT HOME with your computer-manual & forms-$9.95. Write Computer Programs for Profit! How-to guide with forms, letters, tips-$7.95. Also-Computer Consultant Handbook. How to earn $ consulting with business-$7.95. JV Tech, P.O. Box 563, Ludington, MI 49431

Save money shopping via your Home Computer. Use phone modem for automatic shopping. Send $3.00 for catalog and details to:
ZAK, 7787 Stout, Detroit, MI 48228

PROGRAMMERS . . . COPYRIGHT YOUR PROGRAMS. Protect your work. Included: documentation, forms & list of SOFTWARE CO.S all for $7. Blue Cavern Software, 558 J St., Chula Vista, CA 92010

Discount computer printer ribbons for all makes/models Ex. Epson 1500 Nylon $6.99. Catalog. TWS 1314 4th Ave., Coraopolis, PA 15108 (412) 262-1482 Visa or MasterCard.

COMMODORE DUPLICATOR 64
Now you can back-up copy guarded software on the C64.
Our Disk-Based Duplicator is Now Available At A Remarkable Price . . .
$34.95 . . . plus $2.50 shipping.
Major Crdit Cards & C.O.D. orders accepted. We ship within 24 hrs.
DUPLICATING TECHNOLOGIES, INC.,
99 Jericho Tpke., Suite 302A,
Jericho, New York 11753. Daily (516) 333-5808; Eve/Wknds (516) 333-5950

COMPUTE! Classified is a low-cost way to tell over 350,000 microcomputer owners about your product or service.

Rates: $25 per line, minimum of four lines. Any or all of the first line set in capital letters at no charge. Add $15 per line for boldface words, or $50 for the entire ad set in boldface (any number of lines).
Terms: Prepayment is required. Check, money order, American Express, Visa, or MasterCard is accepted. Make checks payable to COMPUTE! Publications.
Form: Ads are subject to publisher's approval and must be either typed or legibly printed. One line equals 40 letters and spaces between words. Please underline words to be set in boldface.
General Information: Advertisers using post office box numbers in their ads must supply permanent address and telephone numbers. Orders will not be acknowledged. Ad will appear in next available issue after receipt.
Closing: 10th of the third month preceding cover date (e.g., June issue closes March 10th). Send order and remittance to: Harry Blair, Classified Manager, COMPUTE!, P.O. Box 5406, Greensboro, NC 27403. To place an ad by phone, call Harry Blair at (919) 275-9809.
Notice: COMPUTE! Publications cannot be responsible for offers or claims of advertisers, but will attempt to screen out misleading or questionable copy.

Advertisers Index

ONLY A FANTASY GAMER COULD CALL THIS HEAVEN.

If exploring eerie dungeons filled with monsters is your idea of fun, we've got two fantasy games that'll have you floating on cloud nine. Each breaks new ground in role-playing games with special features:

WIZARD'S CROWN™ lets you resolve combat two ways: The computer can do it quickly, or you can personally direct it with a multitude of tactical options.

RINGS OF ZILFIN™ adds unprecedented realism to fantasy gaming with its superb graphics. The fully animated scrolling screen grants you step-by-step control of the action.

The gates of heaven are your local computer/software or game store. Enter them today.

If there are no convenient stores near you, VISA & M/C holders can order these $39.95 games by calling toll-free 800-443-0100, x335. To order by mail, send your check to: **STRATEGIC SIMULATIONS, INC.**, 883 Stierlin Road, Building A-200, Mountain View, CA 94043. (California residents, add 7% sales tax.) Please specify computer format and add $2.00 for shipping and handling.

All our games carry a "14-day satisfaction or your money back" guarantee.

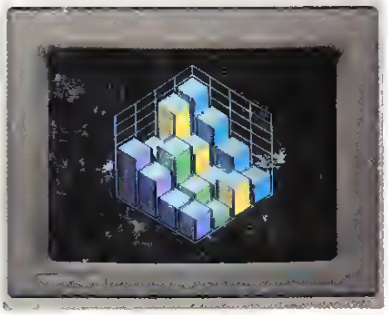
WRITE FOR A FREE COLOR CATALOG OF ALL OUR GAMES TODAY.

ON DISK FOR 48K APPLE® II SERIES AND C-64™.

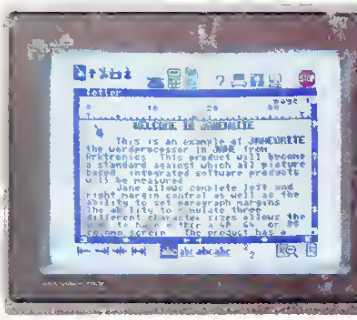
APPLE and COMMODORE 64 are trademarks of Apple Computer, Inc. and Commodore Electronics, Ltd., respectively

RINGS OF ZILFIN includes graphics routines from Penguin Software™'s Graphics Magician®

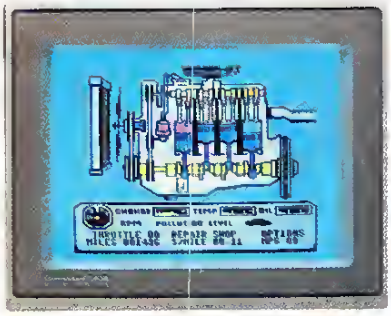
All you need to do this



graph a spreadsheet



write a navel



fix an engine

compose a sang

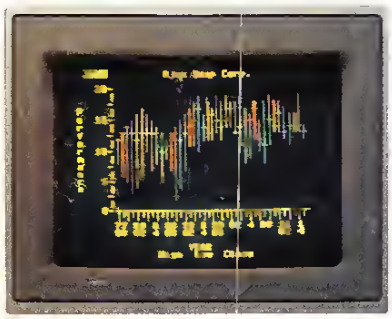
paint a picture

ur banking

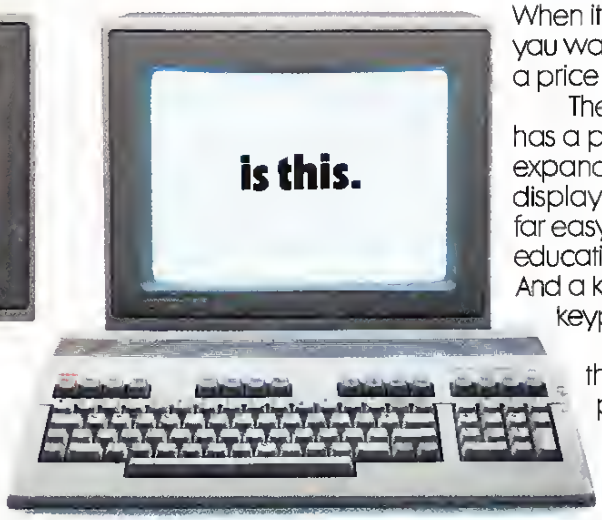
learn ta fly

arganize a data base

tell a stary



farecast sales



When it comes ta persanal computers, yau want the smartest yau can awn. At a price that makes sense.

The new Cammadare 128™ system has a pawerful 128K memary, expandable ta 512K. An 80-calumn display and 64, 128 and CP/M® modes far easy access ta thausands af educatianal, business and hame programs. And a keyboard, with built-in numeric keypad, that aperates with little effart.

Discaver the persanal computer that daes mare far yau. At the price yau've been waiting far. Fram the campany that sells mare persanal computers than IBM® ar Apple.®

COMMODORE 128 PERSONAL COMPUTER

A Higher Intelligence

RETROMAGS

Our goal is to preserve classic video game magazines so that they are not lost permanently.

People interested in helping out in any capacity, please visit us at www.retromags.com.

No profit is made from these scans, nor do we offer anything available from the publishers themselves.

If you come across anyone selling releases from this site, please do not support them and do let us know.

Thank you!